目 录

第1章　MATLAB 系统简介

本章主要介绍 MATLAB 的基础知识。MATLAB 是集数学计算、结果可视化和编程于一身，能够方便地进行科学计算和大量工程运算的数学软件。通过本章的学习，读者将会初步了解 MATLAB 的起源、特点、组成、MATLAB 的安装和卸载以及 MATLAB 帮助文件的使用等知识。

MATLAB 是一套功能十分强大的工程计算及数值分析软件，目前，它已经成为世界上应用最广泛的工程计算软件之一。在美国等发达国家的理工类大学里，MATLAB 是大学生必须掌握的一种基本工具，而在国外的研究设计单位和工业部门，它更是研究和解决工程计算问题的一种标准软件，并被誉为工程技术人员必备软件。因为其简单易用、人机界面良好，又有着演算纸式的科学计算语言的美称。在国内，越来越多的理工科大学生和科学技术工作者正在学习和使用 MATLAB 语言。MATLAB 工具使复杂繁琐的科学计算和编程变得日益简单和准确有效。值得提出的是，MATLAB 不但功能强大而且易学易用，只要掌握一些使用 Windows 操作系统的经验，就可以在较短的时间内掌握其主要内容和基本操作，并运用到工作和学习中去。

1.1　MATLAB 概述

1.1.1　MATLAB 的起源与发展

MATLAB 的最初版本是由 Cleve Moler 博士用 FORTRAN 语言开发的矩阵分析软件，MATLAB 是“矩阵实验室”（MATrix LABoratory）的缩写，它是一种以矩阵运算为基础的交互式程序语言，最早用来作为 LINPACK（线性代数软件包）和 EISPACK（基于特征值计算的软件包）矩阵软件工具包的接口。在 80 年代初期，由 Cleve Moler 和 John Little 采用 C 语言改写了 MATLAB 的内核。不久，他们成立了 MathWorks 软件开发公司，并于 1984 年将 MATLAB 正式推向市场。1992 年初推出了应用于 Windows 操作系统的 MATLAB 4.x 版本，1997 年推出 5.1 版本，1998 年推出 5.2 版本，1999 年推出 MATLAB 5.3 版本，2000 年又推出了更为简便易学的MATLAB 6.0版本。在MATLAB 的版本中，MATLAB 5.3 对应于 Release 11，MATLAB 6.0 则对应于 Release 12。会出现这种版本号和 Release 号不等的情况是因为 Release 对应于 MATLAB 的所有产品（包括各种工具箱），但就 MATLAB 语言来说，它仅是 MATLAB 诸多产品中的一员。因此，我们在后面的章节中都把 MATLAB Release 12 简写成 MATLAB R12。随着 MATLAB 版本的升级，其内容不断扩充和改进，人机界面越来越生动、友好，语言也越来越简单易学，同时，对使用环境也提出了更高的要求。

现在的 MATLAB 已经不仅是用于工程计算的数学软件了，它还包含具有数百个内部核心函数的 MATLAB 主程序和许多功能各异的工具箱（Toolbox）以及 Simulink 系统仿真等功能。MATLAB 以矩阵运算为基础，把矩阵计算、可视化和程序设计融合到一个简单易用的交互式工作环境中，以实现工程计算、算法研究、符号运算、建模和仿真、原型开发、数据分析及可视化、科学和工程绘图、应用程序设计以及图形用户界面设计等工作。

1.1.2 MATLAB 的主要特点

MATLAB 自 1984 年正式推出后，其功能越来越强大，已成为国际公认的最优秀的数学软件之一，尤其是 MATLAB 6.0，其应用范围涵盖了工业、电子、医疗以及建筑等领域。其主要特点大致如下：

- MATLAB 的基本单位为矩阵，其表达式与数学、工程计算中常用的形式类似。并且矩阵的行和列无需定义，可随时添加或修改；
- MATLAB 语言以解释方式工作，对每条语句进行解释后即运行，键入算式即得结果，无需编译，对错误可立即做出反应，大大减少了编程和调试的工作；
- 具有非常友好的人机界面。MATLAB 语言规则与人们长期以来使用的在演算纸上进行演算的书写习惯十分相似，易学易读适于交流；
- 具有强大的作图和数据可视化功能。可以把数据以多种形式加以表现，非常简单、直观、方便；
- 具有极强的可扩展性。MATLAB 软件包括 MATLAB 主程序和许多日益增多的工具箱，工具箱实际就是用 MATLAB 的基本语句编写的各种子程序集，用于解决某一方面的专门问题或实现某一类的新算法。MATLAB 还提供了与其他应用语言的接口，以实现数据的共享和传递。

1.1.3 MATLAB 的基本组成

MATLAB 主要由 MATLAB 主程序、Simulink 动态系统仿真和 MATLAB 工具箱三大部分组成。其中 MATLAB 主程序包括 MATLAB 语言、工作环境、句柄图形、数学函数库和应用程序接口五个部分；Simulink 是用于动态系统仿真的交互式系统，允许用户在屏幕上绘制框图来模拟一个系统，并能动态地控制该系统，目前的 Simulink 可以处理线性、非线性、连续、离散、多变量及多级系统；工具箱实际就是用 MATLAB 的基本语句编写的各种子程序集和函数库，用于解决某一方面的特定问题或实现某一类的新算法，它是开散性的，可以应用也可以根据自己的需要进行扩展。MATLAB 工具箱大致可分为功能性的工具箱和学科性的工具特两类。功能性的工具箱主要用于扩展 MATLAB 的符号计算功能、图形建模仿真功能、文字处理功能和与硬件的实时交互过程，如符号计算工具箱等；学科性的工具箱则具有较强的专业性，用于解决特定的问题，如信号处理工具箱（Signal Processing Toolbox）和通信工具箱（Communication Toolbox）等。

1.1.4 MATLAB 在工程教学中的应用

MATLAB 是一种与数学密切相关的算法语言，其强大的矩阵运算和数值计算能力，使之逐渐成为理工科大学生的热门应用软件之一。信息科技的发展对高等教育的影响深

远，利用计算机手段提高教学效率，能够使学生在使用中掌握计算技能，更能使教师在教学中不再回避复杂的计算问题，从而将问题的分析引向更深的层次。MATLAB 应用于工程教学的计算机辅助教学，使将教师和学生从繁琐重复的数学演算中解脱出来，以使将更多的时间应用于对数学概念和解题思维的思考和训练，使教学的效率大大提高。

现在，以 MATLAB 最基本的矩阵运算为例，求线性方程组 $b=Ax$ 的解，其中 b、x 为向量，A 为矩阵。在 MATLAB 中，无需采用各类繁琐的对方程组进行一步步求解的方法，因为 MATLAB 将各种解法预先做成嵌入式函数，用户在使用时只需简单调用即可。比如求此方程组，只需在 MATLAB 工作窗口中键入“$x=A\backslash b$”。即可求出方程组的解。

MATLAB 的计算结果可视化功能使用起来非常方便，如果改变参数，可直观地得出各参数对结果的影响，能使学习者更加容易理解计算模型的物理意义。

1.2 MATLAB R12 的安装和卸载

1.2.1 对系统的要求

MATLAB 语言的应用与机器类型无关，利用 MATLAB 语言编写的程序可以不用做任何修改直接拷贝到其他机型上使用，不用担心是否能在新的硬件环境下正常运行。鉴于微型计算机在大学校园中的普及，本书仅讨论在 PC 机工作环境下的安装和使用。

MATLAB R12 对系统配置的要求如下：

- 系统的最小配置
  - ◆ 基于 Pentium、Pentium Pro、Pentium II、Pentium III 或者 AMD Athlon 系列处理器；
  - ◆ Microsoft Windows 95、Windows 98、Windows NT 4.0 或者 Windows 2000 操作系统；
  - ◆ 光盘驱动器（CD-ROM drive）供安装软件时使用；
  - ◆ 至少需要 64 MB 内存，推荐 128MB 内存空间；
  - ◆ 所需的硬盘空间和硬盘簇（Cluster）的大小有关，在安装时，安装向导会给出提示；
  - ◆ 8 使以上的显卡和支持 256 色的显示器。
- 其他的推荐配置
  - ◆ Microsoft Windows 支持的图形加速卡、打印机和声卡；
  - ◆ 为了运行 MATLAB 的 Notebook，需提安装 Microsoft Word 7.0 （Office 95）、Microsoft Word 8.0（Office 97）或 Office 2000;
  - ◆ 为了编译自己的 MEX 文件，需要安装下列软件之一：Compaq Visnal FORTRAN 5.0 or 6.1、Microsoft Visual C/C++ version 5.0 or 6.0、Borland C/C++ version 5.0 or 5.02、Borland C++Builder version 3.0 or 4.0 or 5.0、Lcc 2.4（Bundled with MATLAB）；

◆ 为了查看和打印随机附带的 PDF 帮助文件，还需要安装 Adobe Acrobat Reader；

◆ MATLAB 也可以通过 TCP/IP 协议从网络进行安装。

1.2.2　MATLAB 6.0 的安装过程

在 MATLAB R12 中，MATLAB6.0 是其主要的块，MATLAB6.0 的安装即为 MATLAB R12 的安装。下面以一台典型的 PC 机（Window 98 操作系统）为例，介绍 MATLAB R12 的安装过程。

（1）将 MATLAB 6.0 光盘放入光驱，Windows 将会自动运行安装程序。如果没有自动安装，可在资源管理器中光驱盘符的 MATLAB 目录下，用鼠标左键双击 Setup.exe 文件，系统将显示如图 1-1 所示的“Welcome to The MathWorks Installer”（欢迎安装）对话框。

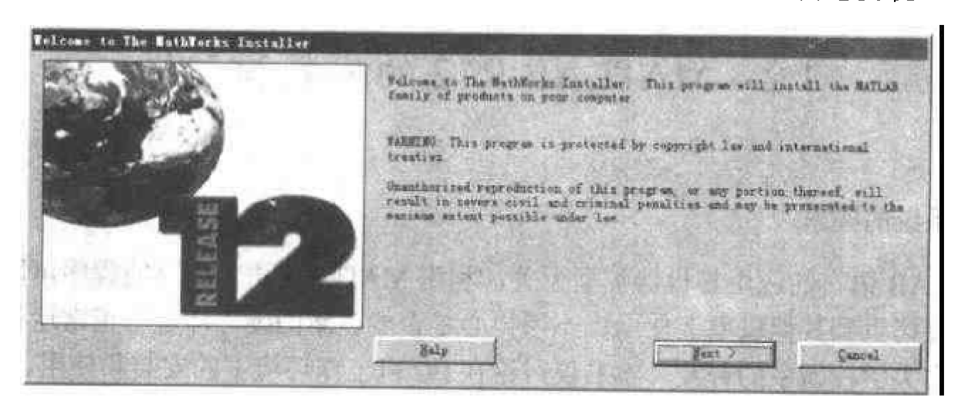


图 1-1　“欢迎安装”对话框

（2）单击【Next】按钮，出现“License Agreement”（软件许可协议）对话框，如图 1-2 所示。

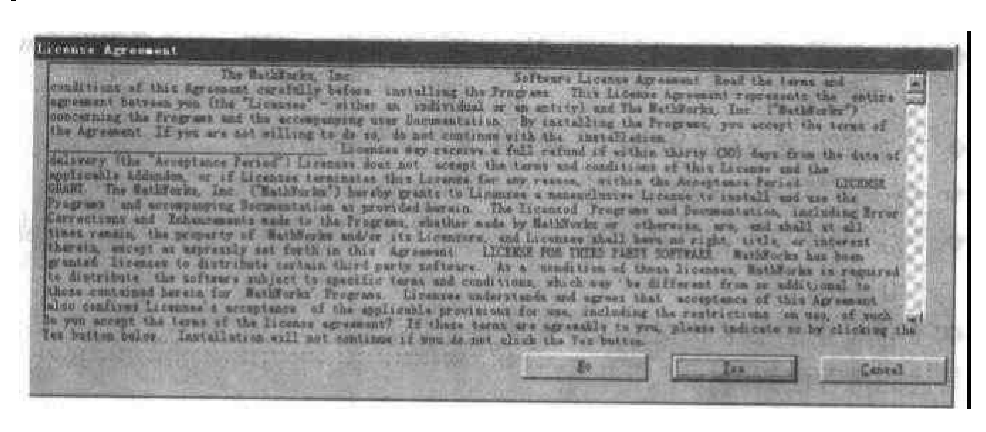


图 1-2　“软件许可协议”对话框

（3）单击【Yes】按钮接受协议内容，出现“Customer Information”（用户信息）对话框，如图 1-3 所示。

（4）按提示输入用户姓名和公司名称。单击【Next】按钮，如果输入序列号正确，便进入如图 1-4 所示的“Product List”（产品列表）对话框，用户可按照提示选择安装路径和安装组件。

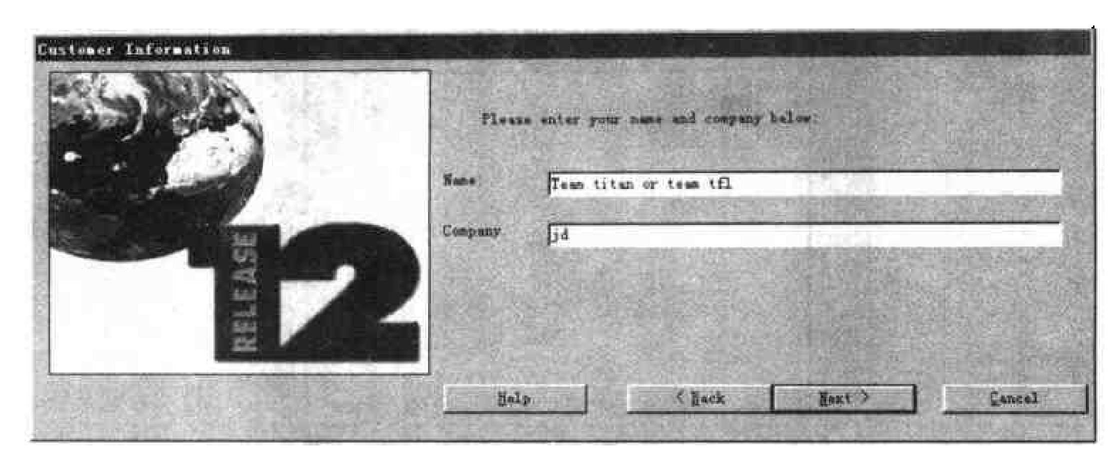


图 1-3 "用户信息"对话框

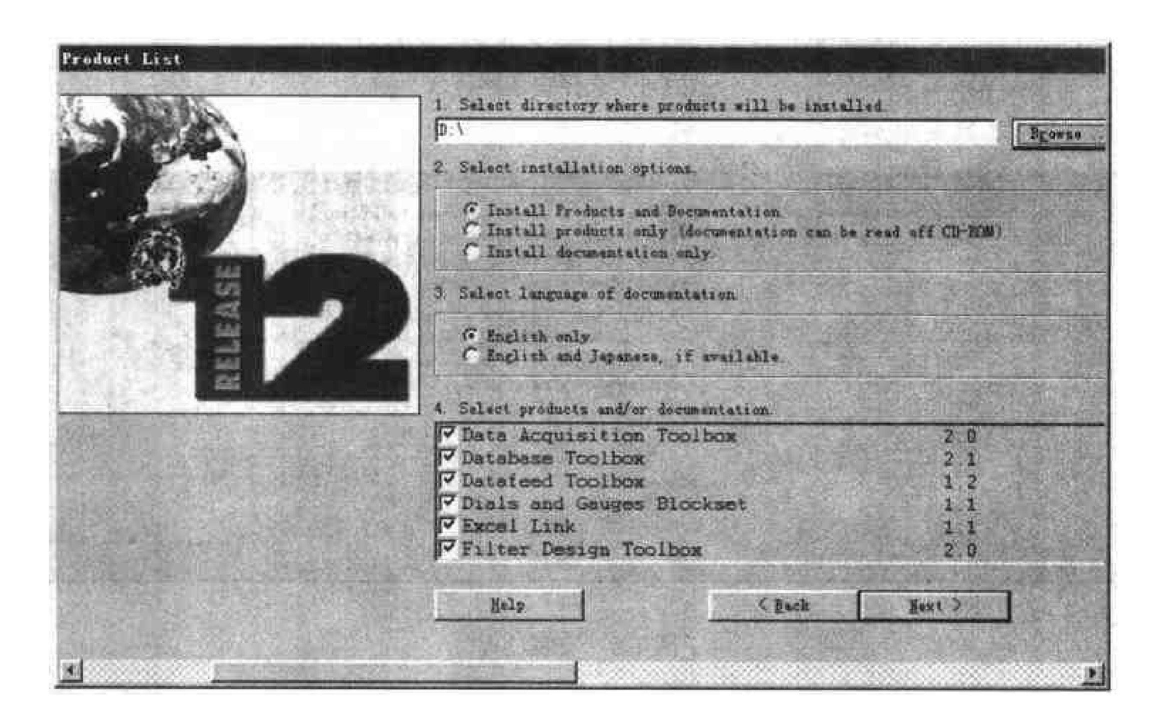


图 1-4 "产品列表"对话框

（5）在对话框中将复选框选中，即选择所有的安装组件。用户可通过单击【Browse】按钮选择 MATLAB 的安装路径，如图 1-5 所示，单击【OK】按钮，确认所选择的安装路径。单击图 1-4 中的【Next】按钮，将出现如图 1-6 所示的安装过程窗口。

（6）安装完毕，出现如图 1-7 所示的对话框。用户可在单选框中选择立即重启动或者稍后重启动计算机。单击【Finish】按钮，完成 MATLAB 的安装。

完成了 MATLAB 的安装过程后，系统将会自动在桌面上创建 MATLAB R12 的快捷方式。

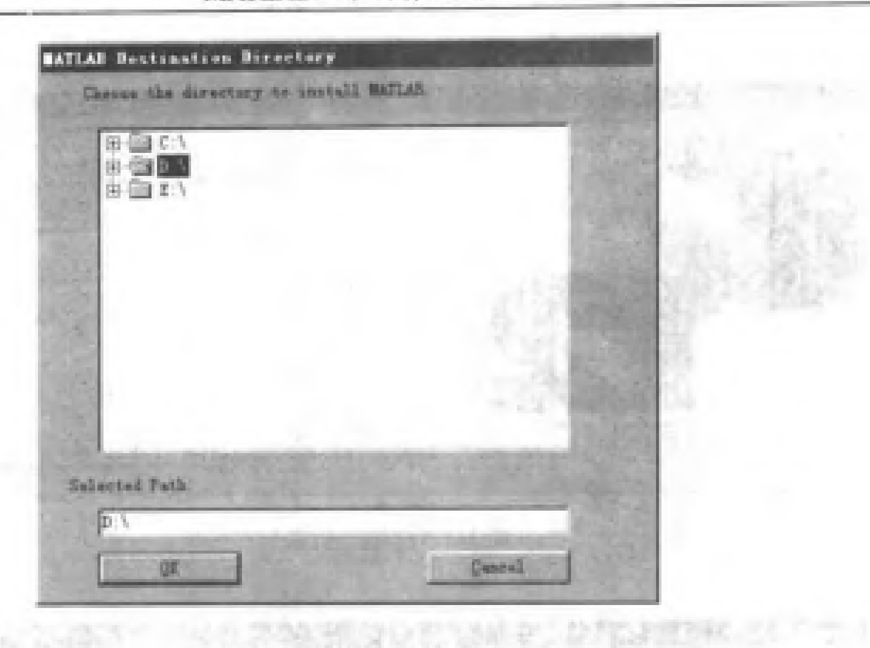


图 1-5 选择安装路径

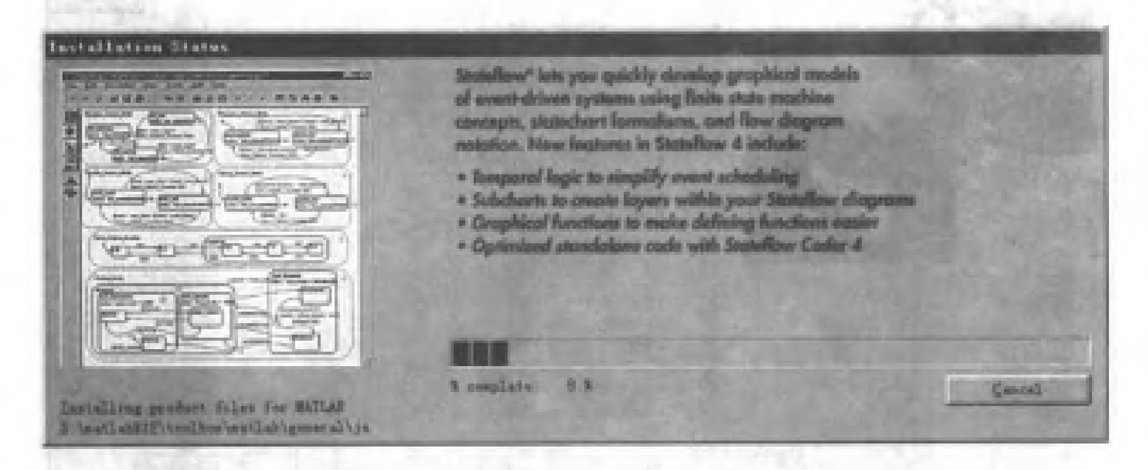


图 1-6 安装过程

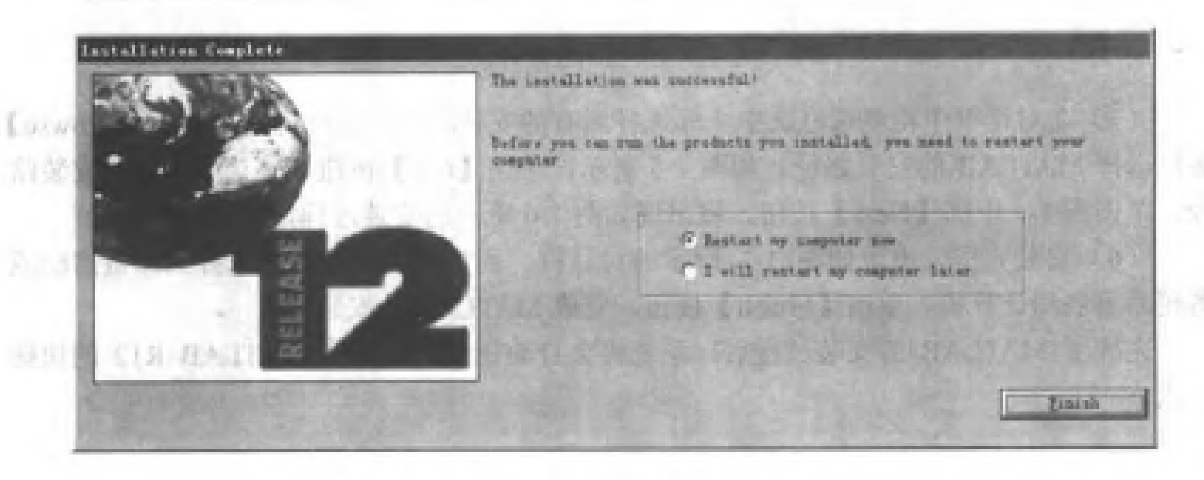


图 1-7 “安装完成”对话框

1.2.3 MATLAB 6.0 的卸载

当用户需要卸载 MATLAB 6.0 时，选择【开始】▸【程序】▸【MATLAB Release12】▸【R12 Uninstaller】菜单命令，将出现如图 1-8 所示“Uninstall Product List”（卸载文件列表）对话框，用户可以选择需要卸载的部分或全部内容。单击【Next】按钮，进行卸载，单击【Cancel】按钮，取消卸载。

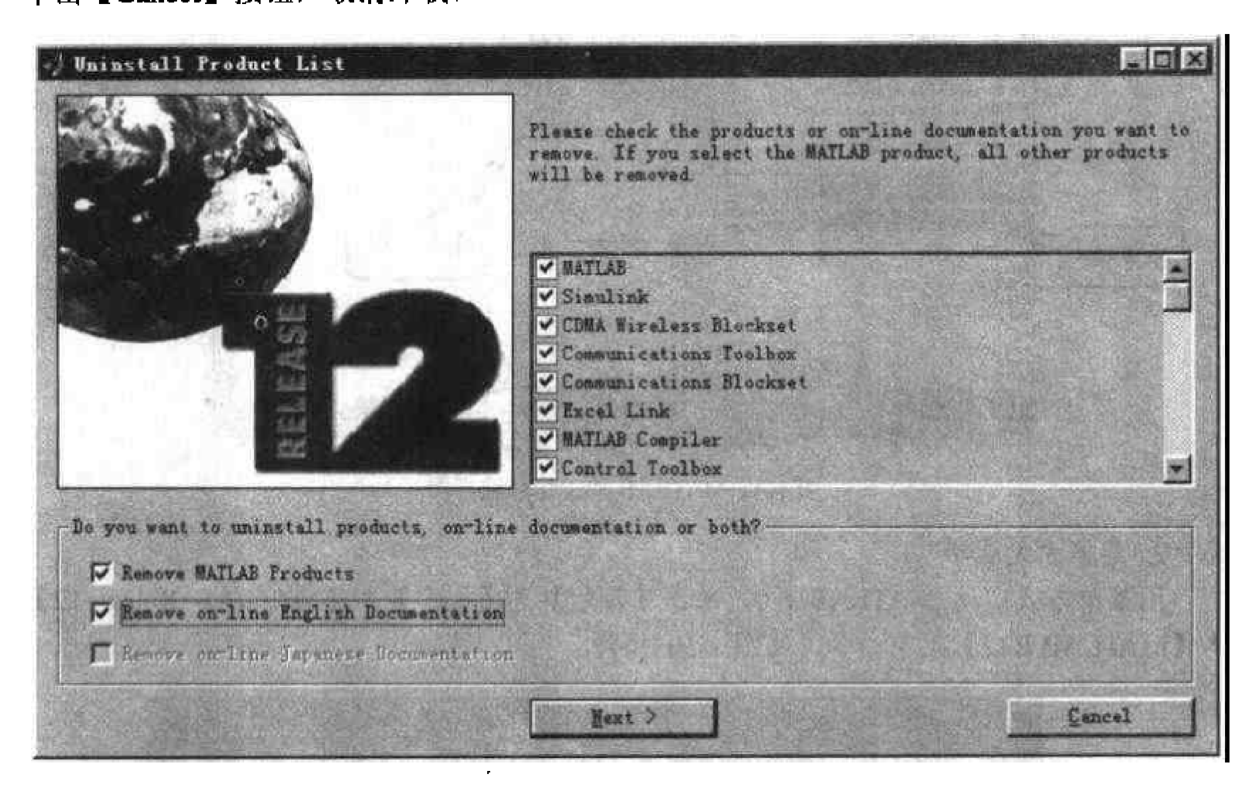


图 1-8 “卸载文件列表”对话框

1.3 MATLAB 快速入门

本节将通过介绍 MATLAB 的工作环境，使初学者初步掌握 MATLAB 软件的基本操作方法。

1.3.1 MATLAB 的启动和退出

MATLAB 可以通过如下方式启动和退出。

1. 快捷方式启动

用快捷方式启动 MATLAB 6.0 的操作步骤如下：

（1）双击 MATLAB R12 快捷方式图标，即可启动 MATLAB。

（2）启动完成后，即出现 MATLAB 的工作环境界面，如图 1-9 所示。

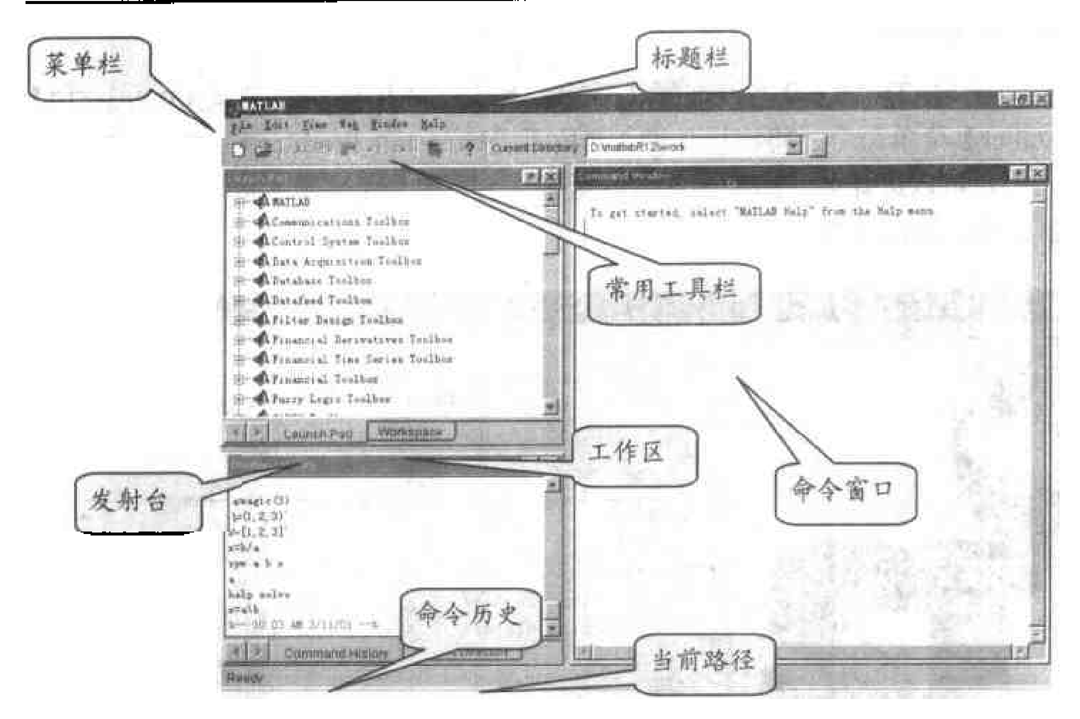


图 1-9 MATLAB 的工作环境界面

2. 菜单方式启动

用菜单方式启动 MATLAB 6.0，单击【开始】▸【程序】▸【MATLAB Release12】▸【MATLAB R12】菜单命令，如图 1-10 所示。

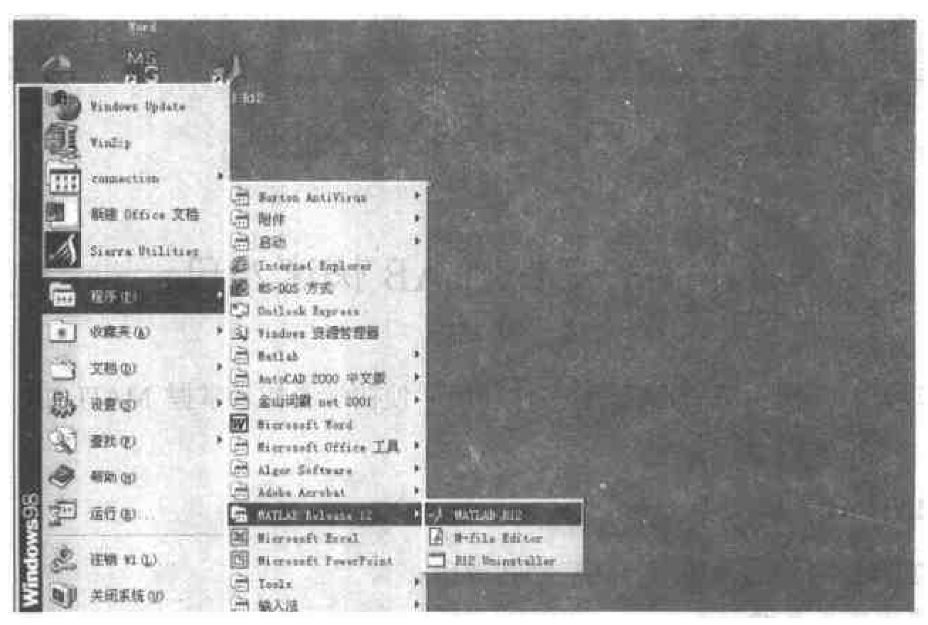


图 1-10 从【开始】菜单进入 MATLAB 操作环境

3. 退出 MATLAB

选择【File】▸【Exit MATLAB】菜单命令或用鼠标单击对话框右上角的关闭按钮，即可退出 MATLAB。

1.3.2　认识 MATLAB 工作环境及操作

MATLAB R12 启动后，产生的工作环境界面（见图 1-9）包含一个工具栏、三个区域、五个工作窗口，这五个工作窗口分别为发射台（Launch Pad）、工作区（Workspace）、命令历史（Command History）、当前路径（Current Directory）和命令窗口（Command Windows）。这是 MATLAB 启动后桌面布置方式的缺省设置。MATLAB 的工作窗口是一个标准的 Windows 界面，可以利用菜单命令完成对工作窗口的操作，使用方法与一般的 Windows 应用程序相同。用户可通过菜单栏【View】的下拉式菜单来控制这些窗口的开启和关闭。下面将通过对菜单栏的逐一介绍来说明这些工作窗口的用法和含义，关于它们的具体用法，将在后续章节中详细介绍。

1. MATLAB 菜单项

MATLAB 菜单栏有【File】、【Edit】、【View】、【Web】、【View】、【Window】以及【Help】六个主菜单选项，用鼠标左键单击即可打开相应菜单，如图 1-11 所示。各菜单的功能分别介绍如下（为简单起见，仅介绍到二级菜单）。

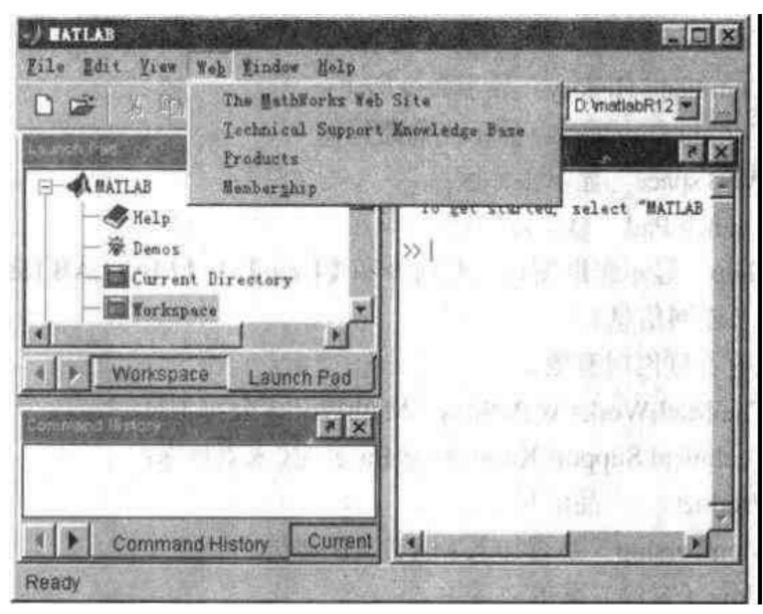


图 1-11　MATLAB 的菜单

- File（文件）菜单
  - ◆ New　新建立一个 M 文件、打开一个新的图形窗口或 Simulink 模型窗口；
  - ◆ Open　弹出“打开文件”对话框，用户可以搜索要打开的已有文件，选中文件，单击【打开】按钮，可以打开一个 M 文件、图形或 Simulink 模型；
  - ◆ Save Workspace as　保存 MATLAB 工作区，用户可任意设置所要保存的文件名和路径；
  - ◆ Set Path　打开路径浏览器；
  - ◆ Preferences　打开“参数设置”对话框，用户可以通过参数设置对话框设置工作环境的外观和操作的相关属性；
  - ◆ Print　打印屏幕内容；

- ◆ Exit MATLAB 退出 MATLAB。
- Edit（编辑）菜单
  - ◆ Undo 撤消上一次的操作；
  - ◆ Cut 将选中的内容剪切，放入剪贴板；
  - ◆ Copy 将选中的内容复制，放入剪贴板；
  - ◆ Paste 将剪贴板中的内容粘贴至指定位置；
  - ◆ Select All 选中命令窗口所有内容；
  - ◆ Delete 删除选中的内容；
  - ◆ Clear Command Window 清除命令窗口中的内容；
  - ◆ Clear Command History 清除命令历史中的内容；
  - ◆ Clear Command Workspace 清除工作区中指定的变量。
- View（视图）菜单
  - ◆ Disktop Layout MATLAB 启动后桌面的布置方式，可有多种选择；
  - ◆ Undock Command Window 将命令窗口单独设置成一独立窗口；
  - ◆ Command Window 显示命令窗口；
  - ◆ Command History 显示命令历史；
  - ◆ Current Directory 显示当前路径；
  - ◆ Workspace 显示工作区；
  - ◆ Launch Pad 显示发射台；
  - ◆ Help 显示帮助窗口。此命令和【Help】▸【MATLAB Help】选项作用相同。
- Web（连网信息）

此菜单只有在联网时有效。

  - ◆ The MathWorks Web Sites MathWorks 公司主页；
  - ◆ Technical Support Knowledge Base 技术支持库；
  - ◆ Products 产品信息；
  - ◆ Membership 是否加入 MATLAB 会员。
- Window（窗口）菜单

当打开多个 MATLAB 窗口时，此菜单可使用户方便地在打开的窗口之间进行切换。

- Help（帮助）菜单
  - ◆ Full Product Pamily Help 显示所有组件的帮助；
  - ◆ MATLAB Help 显示帮助文件；
  - ◆ Using the Desktop 如何使用桌面设置；
  - ◆ Using Launch Pad 如何使用发射台；
  - ◆ Demos 进入 MATLAB 演示窗口；
  - ◆ About MATLAB 显示 MATLAB 版本和用户信息。

2. 工具栏简介

菜单栏的下面是常用工具栏（见图 1-9），上面都是一些常用命令的快捷方式，用鼠标单击命令按钮即可执行相关命令。

工具栏各个按钮的功能如下所述。

- 用 MATLAB 的 M 文件编辑器新建一个 M 文件；
- 用 MATLAB 编辑器打开一个文件；
- 将选中的内容剪切到剪贴板；
- 将选中的内容复制到剪贴板；
- 将剪贴板中的内容粘贴到光标指定的位置；
- 撤消最近一次的操作；
- 打开 Simulink 浏览器；
- 打开 MATLAB 帮助窗口。

此外，在工具栏的右侧，还有一个“当前工作路径”栏，用户在此处设置 MATLAB 的工作路径可以直接键入，也可以通过右边的按钮来设置。

下面通过一个例子来说明 MATLAB 命令输入的基本方法，同时演示 MATLAB 在计算和图形显示方面的强大功能，读者可以上机尝试。

【例 1-1】绘制函数 $y=2\sin(1+x)$的图像，并计算当 $x=0.5$ 时的函数值。

用户只需直接在命令窗口（Command Window）中输入如下内容即可。

```
x=(1:0.2:10);          %给出自变量x的定义域
y=2*sin(1+x);          %写出函数形式
plot(x, y)             %绘出函数图形
y=2*sin(1+0.5)         %求当x=0.5时的y值，其后不加分号，直接在窗口中给出结果
y =1.9950              %输出y的计算结果
```

绘制的函数图形如图 1-12 所示。

说明：在每行输入结束后，按 Enter 键确认。每行后加分号（；）表示不直接在命令窗口中显示本行命令的执行结果，不加分号表示直接输出本行命令的输出结果。“%”在 MATLAB 中为注释符号，其后的部分表示对本行命令的解释。

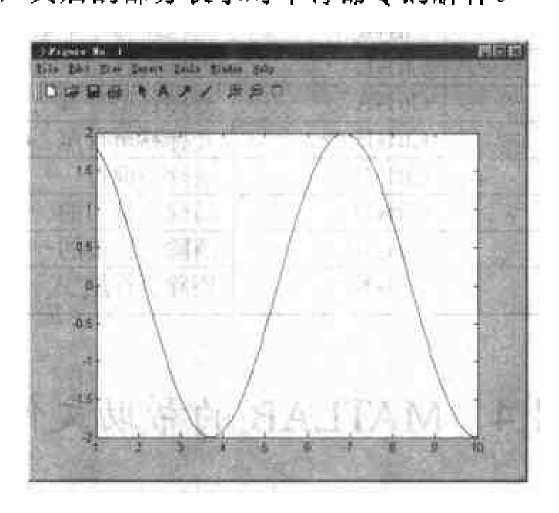


图 1-12 输出的函数图形

1.3.3 MATLAB 通用命令和编辑键

1. 通用命令

在 MATLAB 中，除了可以通过菜单命令对工作窗口进行控制外，用户还可以在

MATLAB 命令窗口中直接键入控制命令并执行。表 1-1 给出了部分常用的通用命令及其功能说明。

表 1-1 部分常用的通用命令

命令名称	功能说明
clear	清除内存中所有的或指定的变量和函数
cd	显示和改变当前工作目录
clc	擦除 MATLAB 工作窗口中所有显示的内容
clf	擦除 MATLAB 当前工作窗口中的图形
dir	列出当前或指定目录下的子目录和文件清单
disp	在运行中显示变量或文字内容
echo	控制运行的文字命令是否显示
hold	控制当前的图形窗口对象是否被刷新
home	擦除命令窗口中所有显示的内容，并把光标移到命令窗口左上角
pack	收集内存碎片以扩大内存空间
quit	关闭并退出 MATLAB
type	显示所指定文件的全部内容
exit	退出 MATLAB

2. 一些常用的编辑键

为便于在 MATLAB 命令窗口中对输入的内容进行编辑，MATLAB 提供了一些控制光标位置和进行简单编辑的常用编辑键和组合键，其命令和用法如表 1-2 所示。

表 1-2 常用的一些编辑键

编辑键	组合键	作用
上箭头	Ctrl+P	调出前一个命令行
下箭头	Ctrl+N	调出后一个命令行
左箭头	Ctrl+B	光标左移一个字符
右箭头	Ctrl+F	光标右移一个字符
Ctrl+左箭头	Ctrl+R	光标左移一个单词
Ctrl+右箭头	Ctrl+L	光标右移一个单词
Home	Ctrl+A	光标移至行首
End	Ctrl+E	光标移至行尾
Esc	Ctrl+U	清除当前行
Delete	Ctrl+D	清除光标后的一个字符
Backspace	Ctrl+H	清除光标前的一个字符
	Ctrl+K	删除至行尾

1.4 MATLAB 的帮助文件

MATLAB 为用户提供了非常详尽的帮助文件，同时提供有多种格式的帮助文件供用户选择，如帮助窗口、HTML 以及 PDF 格式的帮助文件等。此外，MATLAB 还提供了大量的帮助实例和演示供用户使用。通过使用 MATLAB 的帮助菜单或在命令窗口中输入帮助命令，用户可获得所需要的帮助信息，并可以借助帮助文件深入学习 MATLAB。常用的帮助命令如表 1-3 所示。

表 1-3 常用的帮助命令一览表

命令名称	功能说明
help	获得在线帮助
helpwin	打开帮助窗口
demo	运行 MATLAB 演示程序
lookfor	按照指定的关键字查找相关的 M 文件
who	列出当前工作内存中的变量
whos	列出当前工作内存中的变量的详细信息
what	列出当前目录或指定目录下的 M 文件、MAT 文件和 MEX 文件
which	显示指定函数和文件的路径
exist	检查指定名字的变量或文件的存在性
doc	在网络浏览器中显示指定内容的 HTML 格式的帮助文件，或者启动 helpdesk

1.4.1 常用帮助命令

1. help 命令

help 是 MATLAB 最常用的帮助命令，它可以查询所有 MATLAB 函数的用法，并提供绝大多数 MATLAB 命令的使用方法的联机说明。在使用过程中，用户可以随时使用 MATLAB 的 help 命令来获得帮助。help 命令有如下用法：

- 直接输入 help，MATLAB 将列出所有的帮助主题，每个帮助主题对应于 MATLAB 搜索路径中的一个目录。
- help 后加帮助主题，可获得指定帮助主题的帮助信息。

例如，输入 help simulink，将得到 simulink 的帮助信息。

- help 后加函数名，可获得该函数的帮助信息。

例如，要获得正弦函数 sin 的帮助信息，可在命令窗口键入：

help sin %输入内容 help sin

MATLAB 输出：

```
  SIN    Sine.
    SIN（X） is the sine of the elements of X.

Overloaded methods
    help sym/sin.m
```

- help 后加命令名，将得到指定命令的用法。

例如，要获得 what 的帮助信息，可在命令窗口键入：

help what %在 MATLAB 工作窗口中输入 help what 后按 Enter 键，查找 what 的用法

MATLAB 将显示以下内容：

```
WHAT List MATLAB-specific files in directory.
    The command WHAT， by itself， lists the M-files，MAT-files
    and MEX-files in the current working directory.

    The command WHAT DIRNAME lists the files in thirectory dirname on
```

the MATLABPATH. It is not necessary to give the full path name of
the directory; a MATLABPATH relative partial pathname can be
specified instead （see PARTIALPATH）. For example， "what general"
and "what matlab/general" both list the M-files in directory
toolbox/matlab/general.

W = WHAT（'directory'） returns the results of WHAT in a structure
array with the fields:

path	-- path to directory
m	-- cell array of m-file names.
Mat	-- cell array of mat-file names.
Mex	-- cell array of mex-file names.
Mdl	-- cell array of mdl-file names.
P	-- cell array of p-file names.
Classes	-- cell array of class names.

See also DIR，WHO，WHICH，LOOKFOR. %显示与 what 有关的其他命令

2. demo 命令

demo 命令是介绍 MATLAB 功能的演示程序，执行该命令后，系统将显示一个名为“MATLAB Demo Window”的窗口，如图 1-13 所示。

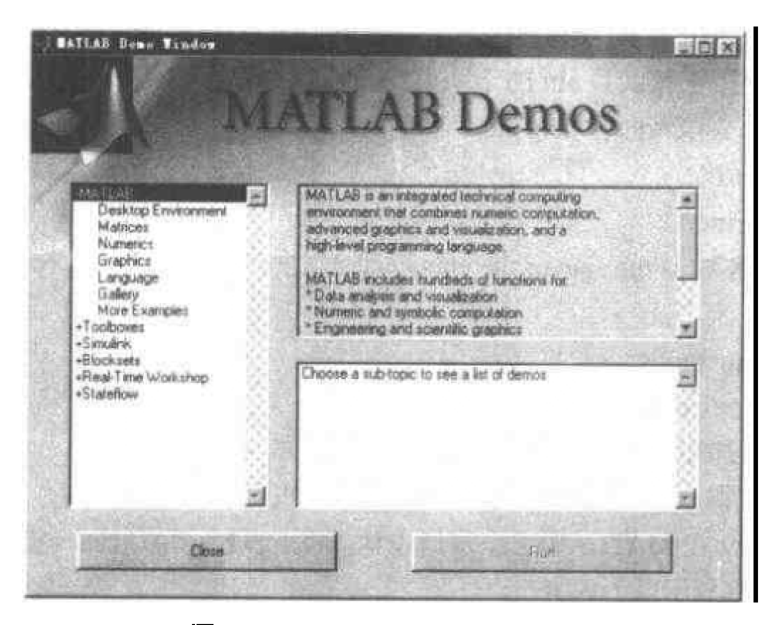


图 1-13 MATLAB 演示窗口

图中有三个文本框，左边是总的结构，在此框中选择需要的内容，右上方的文本框显示对此内容的简介；右下方的文本框显示有关此内容的示例名称，在此文本框中选择要演示的内容，双击相关主题，或选中主题后单击右下方的按钮【RUN XXX】，即可打开此例的演示窗口。

例如，在图 1-13 中选择 MATLAB 下的“Matrices”，右上方文本框同时显示有关“Matrices”的简介，在右下方的文本框中选择要演示的主题，如“Basic matrix operation”（基本矩阵操作），双击该主题或单击【RUN Basic matrix...】按钮，此时 MATLAB 将会打开如图 1-14 所示的演示窗口。

在图 1-14 中，又有多种选择。如果要逐页观察该主题的演示，单击【Start】按钮；也可以单击【AutoPlay】按钮，演示程序将自动播放各页的内容。窗口左下方的文本框给出了每页内容的说明和相应的 MATLAB 程序源代码。单击【Info】按钮将给出此演示程序的相关信息，用户可自行查看。单击【Close】按钮将关闭此演示窗口。

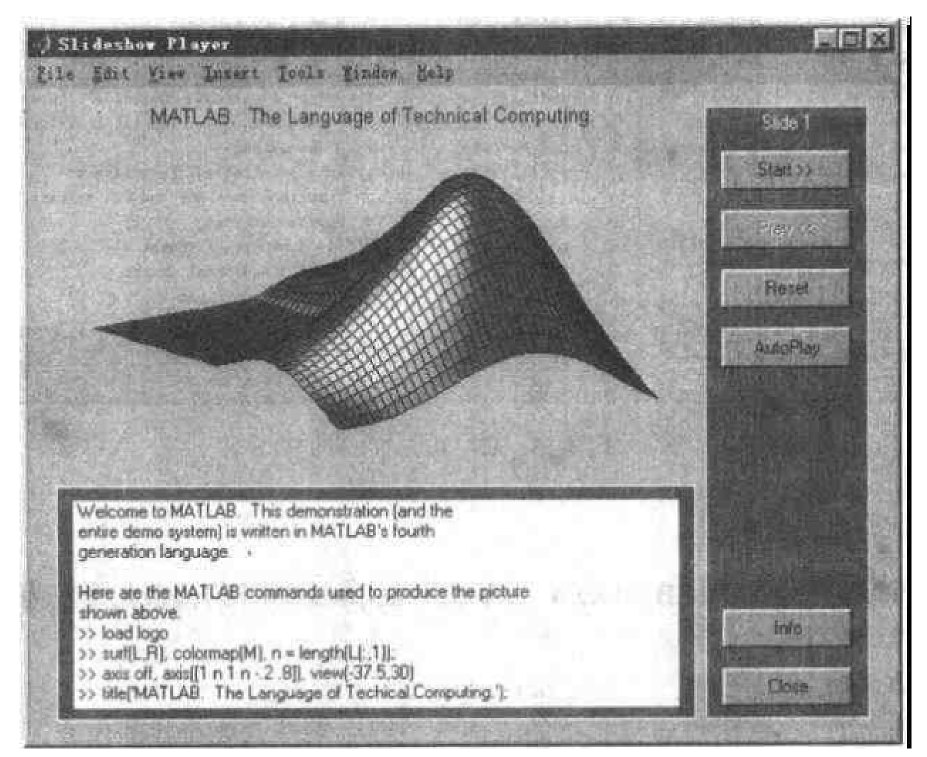


图 1-14 基本矩阵演示窗口

3. helpwin 命令

helpwin 命令用于打开 MATLAB 的帮助文件窗口，在命令窗口中直接键入 helpwin，将出现如图 1-15 所示的帮助窗口。

选择【View】▸【Help】命令，或者【Help】▸【MATLAB Help】选项，可打开同样的帮助窗口，只是所显示的帮助主题起点略有不同。

帮助窗口分为左右两侧，在窗口的上半部为命令区，下半部为内容区。在左侧内容区显示了 MATLAB 所有内容的目录树列表，选中相关项目，则在右侧显示该项目的详细说明。内容区的左侧有四个标签页，分别为 Contents、Index、Search 和 Favorites。单击 Search 标签，在右侧的文本框中输入要查询的命令名或关键字，按【Go】按钮或按 Enter 键，下方的文本框中将会显示相关的帮助内容，右边的列表框中将会显示相关的主题。

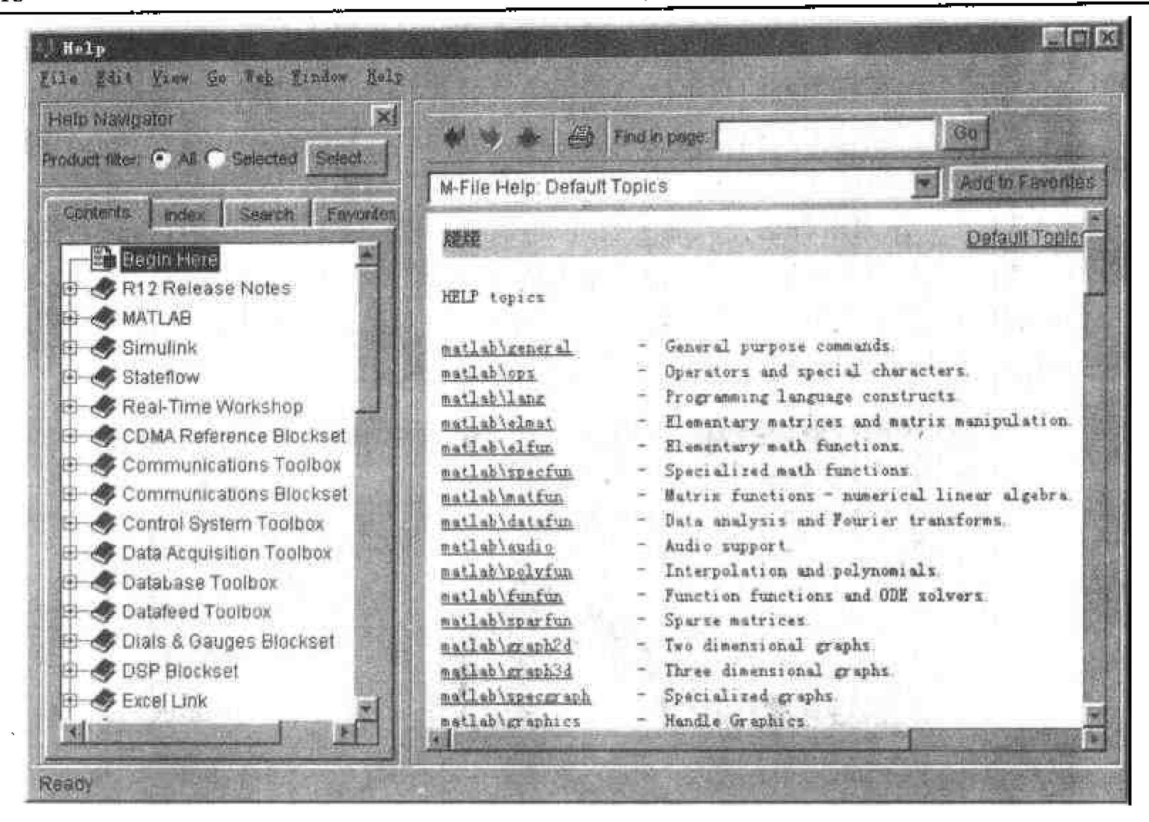


图 1-15 MATLAB 的帮助窗口

1.4.2 其他帮助命令

下面简要介绍 MATLAB 除 help、demo 和 helpwin 以外的其余几个较为常用的帮助命令。

- lookfor

lookfor 命令允许用户通过完整的或部分关键字来搜索要查找的内容，此命令在查找不知道确切名字的具有某种功能的命令或函数时极为有用。通常情况下，lookfor 给出所查主题的帮助文件的第一行信息。

- who 和 whos

who 和 whos 的作用是列出在 MATLAB 工作内存中驻留的变量名，whos 命令同时给出变量的详细信息，例如变量的维数、大小以及所占用的字节和类型。

例如：用户的某一 MATLAB 程序中有 *a*、*b*、*c* 三个变量，现在用 who 和 whos 查询驻留工作内存中的变量情况。

who　　%在 MATLAB 工作窗口中键入 who，并按 Enter 键

MATLAB 将显示：

Your variables are:

a　　　　b　　　　c

whos　　%在 MATLAB 工作窗口中键入 who，并按 Enter 键

MATLAB 的显示内容为：

```
Name      Size          Bytes  Class

a         1×1               8  double array
b         1×1               8  double array
c         4×4             128  double array
```

Grand total is 18 elements using 144 bytes

- exist

exist 命令用来查找或检查变量和函数的存在性，并返回一个 0 到 7 之间的数值。exist 命令的用法和返回值的含义列于表 1-4 和表 1-5 中。

表 1-4 exist 的用法

命令格式	含义
exist（'名字'）	检查指定名字的变量或函数是否存在于 MATLAB 的搜索路径内
exist（'名字'，'var'）	只检查变量
exist（'名字'，'builtin'）	只检查嵌入函数
exist（'名字'，'file'）	检查 MATLAB 的搜索路径内的文件名和目录名
exist（'名字'，'dir'）	只检查目录名

表 1-5 exist 返回值的含义

返回值	含义
0	检查的内容不存在
1	检查的内容是工作内存中的变量
2	检查的内容是 MATLAB 搜索路径内的 M 文件或不知道类型的文件
3	检查的内容是 MATLAB 搜索路径内的 MEX 文件
4	检查的内容是 MATLAB 搜索路径内的 MDL 文件
5	检查的内容是 MATLAB 的嵌入函数
6	检查的内容是 MATLAB 搜索路径内的 P 文件
7	检查的内容是一个目录

1.5 MATLAB 6.0 的新增功能

相对于以前版本，MATLAB 6.0 提供了许多新增功能，主要表现在以下几个方面：

- 工作环境的改进

MATLAB 6.0 工作环境有许多新的变化，主要包括：

- ◆ MATLAB 桌面；
- ◆ 全新的在线帮助；
- ◆ 设置的工具箱通道缓冲减少了 MATLAB 的启动时间；
- ◆ Import Wizard 输入数据向导提供多种数据格式的输入，包括音频和视频格式的数据；
- ◆ 扩展了的环境函数。

● 扩展的数值计算与处理功能

主要包括：

◆ 矩阵计算；
◆ 微分方程求解；
◆ 稀疏矩阵；
◆ 快速傅立叶变换；
◆ 积分；
◆ 函数功能的扩展；
◆ 数据拟合交互界面；
◆ 数据统计交互界面。

● 图形处理方面

图形处理方面主要的新功能有：

◆ 属性编辑器的变化；
◆ 图形输出和打印。

● 三维可视化

MATLAB 6.0 的三维可视化新增加了一些绘制图形的函数，如 Coneplot（绘制三维圆锥体）等。

● 扩展的交互特性和应用程序接口

MATLAB 的新增 Java 界面使用户可以利用 Java 语言的强大功能，通过调用 Java 类创建 Java 对象。

1.6 小　　结

本章主要介绍了 MathWorks 公司的 MATLAB 起源、特点、基本组成部分及其在科学计算等方面的广泛应用，并通过举例特别介绍了 MATLAB 6.0 在工程教学方面的具体使用。通过本章的学习，应该掌握如下内容：

（1）对 MATLAB 语言的起源和发展过程有一定的了解。

（2）掌握 MATLAB R12 的安装和卸载。

（3）掌握 MATLAB 6.0 的两种启动方法，并初步了解 MATLAB 的工作环境及一些简单操作。

（4）掌握 MATLAB 的一些通用命令和编辑键。

（5）学会使用 MATLAB 的帮助命令，并善于利用 MATLAB 强大的帮助资源。

（6）与以往版本相比较，了解 MATLAB 6.0 的新增功能。

习　题

1．按照本章所示安装 MATLAB 6.0，并以两种方式打开 MATLAB 工作窗口，进入 MATLAB 6.0 的工作环境，并退出。

2．尝试、熟悉 MATLAB 6.0 的各栏菜单以及各个工具栏的功能。

3．重新启动 MATLAB 6.0，进入 MATLAB 工作窗口，用 who 命令查看当前工作空间内有无变量及其值。

4．绘制函数 $y=\dfrac{\cos(5x+2)}{\sin(3x+1)}$ 的图像，并求解当 x=2 时的函数值。

5．此时再次用 who 命令查看工作空间内的变量名及其值，与第 3 题比较，同时用 whos 命令查看变量，比较与 who 命令的不同。

6．熟练掌握 MATLAB 的通用命令。

7．练习并熟练掌握 MATLAB 的帮助命令，学会利用 MATLAB 的帮助信息。

8．用 lookfor 命令查找函数 cos 的信息，并与 help 命令查找的结果相比较，注意采用两种命令之间的差别。

第 2 章　MATLAB 的矩阵和数组运算

矩阵和数组是 MATLAB 最基本、最重要的部分，而矩阵和数组运算又是 MATLAB 最基本、最重要的功能。MATLAB 最初是为了矩阵的数值运算而推出的，MATLAB 是分别取 MATrix LABoratory（矩阵实验室）的前三个字母而组成的。MATLAB 之所以能成为世界上主导的计算软件，是因为其具有强大的计算功能。本章将介绍 MATLAB 矩阵和数组的创建、修改及其基本运算。通过本章的学习，读者能够掌握基本的矩阵和数组运算，解决学习和工作中常见的计算问题。

MATLAB 以矩阵为基本的运算单元，向量和标量作为特殊的矩阵处理：向量看作只有一行或一列的矩阵；标量看作只有一个元素的矩阵。这里要注意两点：一是矩阵运算单元定义在复数域上；二是矩阵不必事先定义维数大小，系统会根据用户的输入自动配置，在运算中自动调整矩阵的维数。

2.1　矩阵函数和矩阵运算

2.1.1　矩阵的创建

在 MATLAB 中可以采用多种不同的方法来创建矩阵，本节将逐一介绍矩阵创建的常用方法和特点。

1. 用直接输入法创建矩阵

当需要的矩阵维数比较小时，从键盘上直接输入一系列矩阵元素是最直接、最方便的数值矩阵的创建方法。直接输入法需遵循以下基本规则：

- 整个矩阵应以“[　]”为首尾，即整个输入矩阵必须包含在方括号中；
- 矩阵中，行与行之间必须用分号（；）或 Enter 键（按 Enter 键）符分隔；
- 每行中的元素用逗号（，）或空格分隔；
- 矩阵中的元素可以是数字或表达式，但表达式中不可包含未知的变量，MATLAB 用表达式的值为该位置的矩阵元素赋值。当矩阵中没有任何元素时，该矩阵被称作“空阵”（Empty Matrix）。

【例 2-1】用直接输入法创建一个 4×4 的 A 矩阵。

只要在 MATLAB 工作窗口中直接输入下列矩阵元素即可：

```
A=[2 3 4 5; 3 4 5 6; 4 5 6 7; 7 8 9 10]    %键盘输入内容，元素之间用空格分开
A =                                         %按 Enter 键后显示内容
     2     3     4     5
     3     4     5     6
```

```
    4    5    6    7
    7    8    9    10
```

或者，

```
A = [2 3 4 5
    3 4 5 6
    4 5 6 7
    7 8 9 10]                %键盘输入内容，行与行之间用 Enter 键分隔
A =                          %按 Enter 键后显示内容
    2    3    4    5
    3    4    5    6
    4    5    6    7
    7    8    9    10
```

【例 2-2】用直接输入法创建一个具有表达式元素的 3×3 的 A 矩阵。

```
x=2;y=pi/3;                          %定义变量 x，y 并赋值
A=[2,5,cos(y);x,2*x,3*x;x,x/2,x/4]   %创建矩阵 A，行与行之间用分号隔开
A =                                  %MATLAB 显示内容
    2.0000    5.0000    0.5000
    2.0000    4.0000    6.0000
    2.0000    1.0000    0.5000
```

一旦创建了矩阵，它将被自动存储在 MATLAB 工作空间中，可以使用矩阵名“A”来调用。矩阵中的元素可以用它的行和列表示，如 A(3,2)表示矩阵 A 的第三行第二列的元素。可以用对矩阵元素直接赋值的方法对矩阵进行修改。

提示：由于 MATLAB 的矩阵定义在复数域上，因此矩阵元素可以是复数。通常用书写复数的方法输入复数元素，或者用一个矩阵表示复数矩阵的实部，用另一个矩阵表示复数矩阵的虚部，最后将两个矩阵相加，即可得到所需的复数矩阵。

【例 2-3】分别用直接输入法和矩阵相加法创建一个复数矩阵。

方法一：直接输入法

```
A=[2+3i, 3+4i, 4 ; 1+2i, 2+3i, 5; 7, 2+3i, 9]   %直接输入复数元素
A=                                             %得到复数矩阵 A
    2.0000 + 3.0000i    3.0000 + 4.0000i    4.0000
    1.0000 + 2.0000i    2.0000 + 3.0000i    5.0000
              7.0000    2.0000 + 3.0000i    9.0000
```

方法二：矩阵相加法

```
B=[2,3,4; 1,2,5; 7,2,9];      %创建复数矩阵 A 的实部矩阵
C=[3,4,0; 2,3,0 ;0, 3,0];     %创建复数矩阵 A 的虚部矩阵
A=B+C*I                       %实部矩阵加上虚部矩阵乘 i 得到复数矩阵 A
A =
    2.0000 + 3.0000i    3.0000 + 4.0000i    4.0000
```

1.0000 + 2.0000i　　2.0000 + 3.0000i　　5.0000

7.0000　　2.0000 + 3.0000i　　9.0000

2. 由矩阵编辑器创建和修改矩阵

当需要建立的矩阵很大，不适合手工直接输入时，可以使用矩阵编辑器（Matrix Editor）来创建和修改。其操作步骤如下。

（1）预先定义变量

调用矩阵编辑器之前，需要预先定义一个变量，数值变量和矩阵变量均可。

```
A=[5,6,7; 8,9,10; 2,3,4]                %定义并创建一个矩阵变量A
```

（2）打开工作空间浏览器

选择【View】▸【Workspace】菜单命令，系统将出现如图 2-1 所示“工作区”窗口。窗口内显示此刻 MATLAB 系统内所有的变量。

Workspace

File Edit View Web Window Help

Stack: Base

Name	Size	Bytes	Class
A	3x3	72	double array
a	1x3	24	double array
ans	0x0	0	char array
x	1x451	3608	double array
y	1x1	8	double array

Ready

图 2-1　工作空间浏览器窗口

（3）打开矩阵编辑器

选中变量 *A*，此时工具栏上的和按钮（打开和删除）变为可选，表示可进行此项操作。单击按钮，即可打开 *A*，同时启动矩阵编辑器（Array Editor）。

注意：Array Editor 是 MATLAB 6.0 新增功能，它与以往版本的界面略有不同，如图 2-2 所示。

（4）改变矩阵元素值

用户可以通过鼠标选中图 2-1 中左上方文本框中的矩阵元素，输入要改变的值即可。输入的值可以是数值，也可以是表达式，MATLAB 会自动计算输入表达式的值。本例中所选矩阵为 *A* 的第二行第二列的元素 *A* (2,2)，输入新的元素 sin (9)。

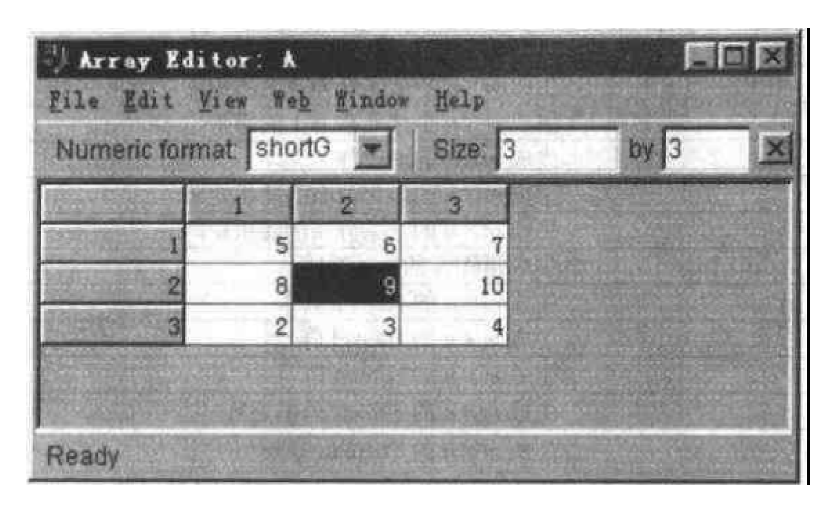


图 2-2　矩阵编辑器窗口

（5）改变矩阵的维数

左下角的两个文本框分别表示矩阵的行和列，此矩阵 *A* 的维数是 3×3，用户可以通过输入数值来改变矩阵 *A* 的行和列。此项功能可以方便地创建和修改矩阵，既可以将原来的矩阵扩展为更大的矩阵，又可以将原来的矩阵裁剪为它的左上方的子矩阵。

（6）确认所有元素符合要求后，关闭该对话框

这时，便创建了一个新矩阵 *A*。检查此时矩阵 *A* 的值：

```
A                  %在工作窗口中键入A
A=                 %MATLAB 输出内容
    5.0000    6.0000     7.0000
    8.0000    0.4121    10.0000
    2.0000    3.0000     4.0000
```

可以发现 *A*(2,2) 已经由原来的 9 改变为 sin(9)，即 0.4121。

3. 由函数创建和修改矩阵

MATLAB 提供了大量的函数用于创建一些特殊的矩阵及其派生矩阵，表 2-1 列出了一些常用函数的含义。

下面将重点介绍矩阵函数 zeros、ones、eye、rand 的调用格式及其应用，其他函数的调用格式与此大致相同，读者可查阅 MATLAB 的联机帮助。

- zeros 生成全部元素为 0 的零矩阵（Zeros Array）

  A=zeros(*n*)　生成 *n*×*n* 零矩阵。

  A=zeros(*m*, *n*)或者 zeros([*m*, *n*])　生成 *m*×*n* 的零矩阵。

  A=zeros(*m*, *n*, *p*,…)或者 zeros([*m n p* …])　生成 *m*×*n*×*p*×…的零矩阵。

  B=zeros(size(*A*))　生成和矩阵 *A* 大小相等的全零矩阵。

表 2-1 一些常用的矩阵生成函数

函数名称	含义和功能
zeros (m, n)	生成 $m \times n$ 全部元素为 0 的零矩阵
ones (m, n)	生成 $m \times n$ 全部元素为 1 的矩阵
eye (m, n)	生成 $m \times n$ 的单位阵
rand (m, n)	生成 $m \times n$ 的均匀分布的随机矩阵
randn (m, n)	生成 $m \times n$ 的正态分布的随机矩阵
company (A)	生成矩阵 A 的伴随矩阵
hadamard (m, n)	生成 $m \times n$ 的 Hadamard 阵
hankel (m, n)	生成 $m \times n$ 的 Hankel 阵
hilb(m, n)	生成 $m \times n$ 的 Hilbert 阵
Invhilb (m, n)	生成 $m \times n$ 的 Hilbert 阵的逆阵
toeplitz (m, n)	生成 $m \times n$ 的 Toeplitz 矩阵
wilkinson (m, n)	生成 $m \times n$ 的 Wilkinson 特征值检验矩阵
Gallery	生成 Higham 的检验矩阵
magic (m, n)	生成 $m \times n$ 的魔方阵
kron(A,B)	生成矩阵 A 和 B 的 Kronecher 张量积
pascal(n)	生成 n 维 Pascal 阵
vander(A)	由矩阵 A 生成 Vander 矩阵

- ones 生成全部元素为 1 的矩阵

  A=ones(n)　生成 $n \times n$ 全 1 矩阵。

  A=ones(m, n)或者 ones([m, n])　生成 $m \times n$ 的全 1 矩阵。

  A=ones(m, n, p,…)或者 ones([m n p …])　生成 $m \times n \times p \times \cdots$ 的全 1 矩阵。

  B=ones(size(A))　生成和矩阵 A 大小相等的全 1 矩阵。

- eye 生成单位阵

  A=eye(n)　生成 $n \times n$ 单位阵。

  A=eye(m, n)或者 eye([m, n])　生成 $m \times n$ 的单位矩阵。

  B=eye(size(A))　生成和矩阵 A 大小相等的单位矩阵。

注意：对于多维数组没有定义单位矩阵，像 ones(m, n, p,…)或者 ones([m n p …])一样的命令 A=eye([m, n, p])MATLAB 将给出出错信息。

- rand 生成均匀分布的随机矩阵

  A=rand(n)　生成 $n \times n$ 随机矩阵。

  A=rand(m, n)或者 rand([m, n])　生成 $m \times n$ 的随机矩阵。

  A=rand(m, n, p,…)或者 rand([m n p …])　生成 $m \times n \times p \times \cdots$ 的随机矩阵。

  B=rand(size(A))　生成和矩阵 A 大小相等的随机矩阵。

  A=rand　不带任何参数将产生一个随机数。

提示：rand 函数产生一个矩阵元素在 0 和 1 之间均匀分布的随机数的随机矩阵。

- diag 生成一个对角阵或由对角线元素组成的向量

  A=diag(V)　当 V 为 n 维向量时，产生一个以向量 V 的元素为对角线的 n 维数组。

  A=diag(V)　当 V 为 n 维矩阵时，产生一个以 V 矩阵的主对角线元素为元素的 n 维数组。

技巧：在 MATLAB 中，不需要事先定义矩阵的维数，MATLAB 自动为矩阵分配存储空间。但如果在程序运行过程中采用零矩阵为矩阵生成的全部元素，或某一行、某一列的元素预先分配内存空间，将会大大加快 MATLAB 程序的运算速度。

【例 2-4】利用 diag 产生对角阵及对角线向量。

```
A=rand(4,4);          %创建一个 4 阶随机矩阵
B=diag(A);
C=diag(B);            %在 MATLAB 工作窗口中输入以上内容
A                     %输入 A 按 Enter 键，表示显示变量 A 的内容
A =
    0.7833    0.7942    0.4154    0.7680
    0.6808    0.0592    0.3050    0.9708
    0.4611    0.6029    0.8744    0.9901
    0.5678    0.0503    0.0150    0.7889
B                     %输入 B 按 Enter 键
B =
    0.7833
    0.0592
    0.8744
    0.7889
C                     %输入 C 按 Enter 键
C =
    0.7833         0         0         0
         0    0.0592         0         0
         0         0    0.8744         0
```

提示：MATLAB 程序中的所有内容均需用英文输入，并且变量区分大小写。否则，MATLA 会给出出错信息。

4. 从外部数据文件调入矩阵

在 MATLAB 中还可以从外部数据文件中读入数据生成矩阵。这些外部数据文件可以是以前 MATLAB 生成的矩阵存储的二进制文件，也可以是包含数值数据的文本文件。在文本文件中，数据要排成一个矩形表，数据之间用空格分开，数据的每行仅包含矩阵的一行，并且各行的元素个数必须相等，也就是说，在文本文件中的数据在调入之前要预先排列成矩阵的形式。

调入方法：

```
load filename.dat 或 filename.txt   %将数据文件 filename 的数据内容调入工作空间
filename                            %显示以 filename 命名的矩阵变量的内容
```

另外，MATLAB 6.0 新增加了一个 Import Wizard 功能，可从外部数据文件中直接读取数据。利用此项功能可方便地读出数据文件中的数据，其使用方法如下：

（1）选择【File】▸【Import Data...】命令选项，将出现如图 2-3 所示对话框，选择将要读取数据的数据文件。选中文件，单击【打开】按钮，然后打开如图 2-4 所示"Import

Wizard”窗口。

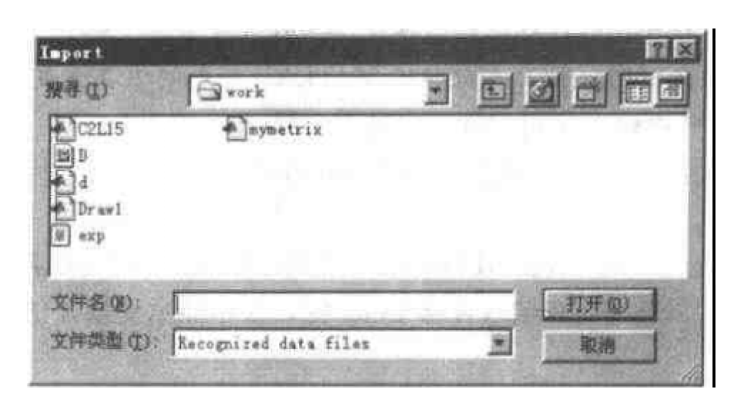


图 2-3 选择要打开的数据文件

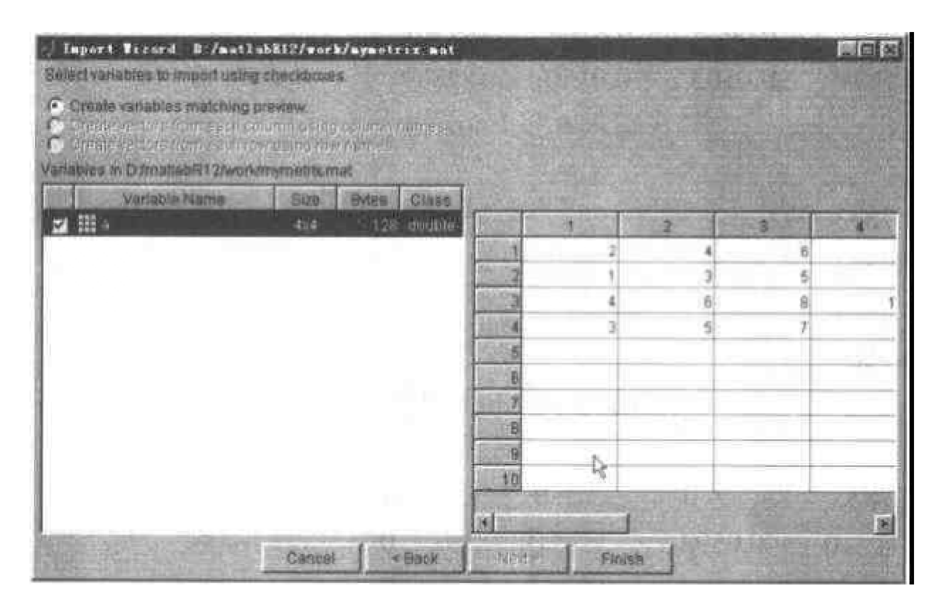


图 2-4 “Import Wizard”窗口

（2）单击【Finish】按钮，即可输入所打开的数据文件中的数据。例如在图 2-4 中，打开 D:/MATLABR12/work/mymetrix.mat 文件，文件中包含一个矩阵变量 a。单击【Finish】按钮后，在 MATLAB 命令窗口中键入 a，系统则输出：

```
a =
     2     4     6     8
     1     3     5     7
     4     6     8    10
     3     5     7     9
```

这说明，此时矩阵 a 已经在 MATLAB 的工作环境中（也可通过查看工作区变量来显示矩阵 a 是否存在）。

【例 2-5】由外部数据文件直接读入数据生成一个矩阵。

（1）用写字板或笔记本或其他文本编辑器，创建一个包含如下数据的数据文件，命名为 shuju.txt。

```
2.0   4.0   5.0   7.0
6.0   3.0   8.0   10.0
25.0  21.0  13.0  12.0
16.0  14.0  19.0  17.0
```

（2）用下述命令将 shuju.txt 中的内容调入工作空间并生成矩阵变量 shuju。

```
load shuju.txt
shuju                %显示矩阵变量 shuju
shuju =
         2     4     5     7
         6     3     8    10
        25    21    13    12
        16    14    19    17
```

技巧：采用此方法创建和保存的矩阵大小没有任何限制，这样用户即可将其他程序生成的数据文件直接调入 MATLAB 中使用。

注意：在文本文件中，数据要排成一个矩形表，数据之间用空格分开，数据的每行仅包括矩阵的一行，并且各行的元素个数必须相等，即在文本文件中的数据在调入之前要预先排列成矩阵的形式。

2.1.2 矩阵的保存和提取

MATLAB 用 Mat 文件保存二进制的数据。如果有大量的矩阵或较为重要的数据需要用矩阵的形式保存和再次使用时，就需要进行矩阵的保存和提取。可以用 Mat 文件来保存和提取矩阵。

用【Save】命令保存已经存在的矩阵 *A* 和 *B*。其调用格式为：

save filename A B

filename 是用户自己定义的文件名，MATLAB 系统将会自动在文件名后加上后缀.MAT。MATLAB 系统默认的路径是 MATLABR11\bin，如果用户要改变路径，可以在文件名前面加上路径，或在路径浏览器里改变路径。

重新启动 MATLAB 后，可以用 Load 命令将保存在文件中的矩阵读到 MATLAB 工作空间的内存中来。其调用格式为：

load filename

【Load】命令不能指定变量名，系统仍然将 *A*、*B* 作为矩阵的名称。用户可以用 who 或 whose 来查看 MATLAB 工作空间内存中的变量。

2.1.3 矩阵元素的标识

矩阵是 MATLAB 的基本运算单元，向量和标量都作为特殊的矩阵处理：向量看作只有一行或一列的矩阵；标量看作只有一个元素的矩阵。因此矩阵的子矩阵就可通过向量、标量的标识来引用和赋值。此外，还可通过 MATLAB 的冒号运算来应用。下面将对向量的生成作一定的阐述，然后再具体说明矩阵的子引用和赋值方法。

1. 向量的生成

MATLAB 有多种方法生成向量，除了在命令窗口中直接输入之外，本书着重介绍最常用的三种自动创建向量的方法，即利用冒号、linspace 函数和 logspace 函数生成向量。

- 利用冒号生成向量

冒号具有多种功能，在用于生成向量时，使用以下格式：

x=i:j

- ◆ 如果 $i<j$，生成均匀等分向量 $x=[i,i+1,i+2,\ldots j]$；
- ◆ 如果 $i>j$，则生成 x 空向量。

x=i:j:k

- ◆ 如果 $j>0$ 且 $i<k$ 或 $j<0$ 且 $i>k$，则生成向量 $x=[i, i+j, i+2j, \ldots, k]$；
- ◆ 如果 $j>0$ 且 $i>k$ 或 $j<0$ 且 $i<k$，则生成向量 x 为空向量。

【例 2-6】利用冒号生成向量。

```
a=1:6;
b=2:2:10;
c=8:-1:2;
c1=8:1:4;
a                    %显示变量a的内容
a =
    1     2     3     4     5     6
b
b =
    2     4     6     8    10
c
c =
    8     7     6     5     4     3     2
c1
c1 =
     Empty Matrix: 1-by-0
```

这说明 $c1$ 生成的是空向量。

- 利用 linspace 生成向量

linspace 函数生成等差向量，其功能类似于冒号算子 $x=i:j$，调用格式如下：

x=linspace(a,b)

生成有 100 个元素的行向量 x，其元素值在 a、b 之间线性分布。

x=linspace(a,b,n)

生成有 n 个元素的行向量 x，其元素值在 a、b 之间线性分布。

- 利用 logspace 生成向量

logspace 生成等比向量，其用法与 linspace 相同。

x=logspace(a,b)

生成有 50 个元素的行向量 x，其元素起点 $x(1)=10^a$，终点 $x(50)=10^b$。

***x*=logspace(*a*,*b*,*n*)**

生成有 n 个元素的行向量 x，其元素起点 $x(1)=10^a$，终点 $x(50)=10^b$。

【例 2-7】利用 linspace 函数生成等差向量，利用 logspace 函数生成等比向量。

```
a=linspace(1.5,6.0,5);   %元素值在 1.5 和 6.0 之间的 5 个数
b=logspace(0,2,4);       %元素值在 100 和 102 之间的 4 个数
a
a =
    1.5000    2.6250    3.7500    4.8750    6.0000
b
b =
      1.0000    4.6416    21.5443    100.0000
```

提示：linspace 函数和冒号算子的使用不同。其不同之处在于冒号算子根据给定的初始值、增量和终止值控制生成向量元素的个数，而 linspace 函数给定元素的个数，根据初始值和终止值确定向量各个元素的具体值。

2. 矩阵的标识

矩阵中的元素或子矩阵可以用标量、向量和冒号的标识来引用和赋值。常用以下两种方法：

- 子矩阵的序号向量标识方式 $A(u, v)$

u,v 可以是任意排列的向量，其中任何一个可以是冒号，它表示全部行（在 u 的位置）或全部列（在 v 的位置）。

注意：用户最好使用序号单调的向量 u、v，否则有可能产生混乱。

- "0~1" 向量标识方式 $A(L1, :)$、$A(:, L2)$和 $A(L1, L2)$

$L1$、$L2$ 为向量，其长度分别为矩阵 A 的行数或列数；向量 $L1$、$L2$ 的元素或者取 1，或者取 0，它是逻辑数组（Logical Array）。"1" 表示 "是" 并提取相应的行或列；"0" 表示 "否"，不提取元素。此种表示方法主要是为了与关系运算符配合使用，在寻找矩阵中某行或某列中所有大于或者小于某数的元素时特别有用。

提示："0~1" 向量标识方式的原理是：通过与 A 同样大小的逻辑值向量 L 中 "逻辑值 1" 所在的位置来标识出矩阵 A 中符合条件元素的相应位置。

【例 2-8】用序号标识方式提取矩阵 A 的子矩阵。

```
A=magic(4);               %产生 4 维的魔方阵
A1=A(1:2,[1,2,4]);        %提取矩阵 A 的第一行和第二行的第 1，2，4 个元素
A2=A([4,1],:);            %提取矩阵 A 的第四行和第一行的全部元素
A([1,3],[2,4])=zeros(2);  %使矩阵第一行和第三行的第 2，4 个元素为 0
A
A =
    16     0     3     0
     5    11    10     8
     9     0     6     0
     4    14    15     1
```

```
A1
A1 =
    16     2    13
     5    11     8
A2
A2 =
     4    14    15     1
    16     2     3    13
A
A =
    16     0     3     0
     5    11    10     8
     9     0     6     0
     4    14    15     1
```

【例 2-9】用“0~1”标识法标识并提取矩阵 A 中符合条件的元素。

```
A=zeros(4,4);        %预先生成一个 4*4 的零矩阵，以便为矩阵 A 预留内存空间
A(:)=-8:7            %为矩阵 A 运用“全元素法赋值”，在-8 和 7 之间等差分布
A =
    -8    -4     0     4
    -7    -3     1     5
    -6    -2     2     6
    -5    -1     3     7
L=abs(A)>4;          %列出矩阵 A 中所有绝对值大于 4 的元素位置，产生与 A 同维
                     数的“0~1”逻辑矩阵 L
L
L =
     1     0     0     0
     1     0     0     1
     1     0     0     1
     1     0     0     1
X=A(L);              %将 A 中被矩阵 L 标识的所有元素赋给向量 X
X
X =
    -8
    -7
    -6
    -5
     5
     6
```

7

由例 2-9 可以看出，矩阵 *A* 中所有绝对值大于 4 的元素全部列于向量 *X* 中。可见，用"0~1"向量标识法可以方便地取出矩阵中所有符合条件的元素，并可以通过逻辑矩阵清楚地标明该元素所在的位置。

2.1.4 基本矩阵函数和矩阵分解函数

在 MATLAB 中可以通过已知矩阵的旋转、截取等变换来得到用户所需的新矩阵，MATLAB 提供的矩阵变换命令和函数如表 2-2 所示。

表 2-2 矩阵变换函数表

函数名		含义和功能
矩阵旋转函数	*B*=fliplr(*A*)	将 *A* 矩阵左右翻转得到 *B* 矩阵
	B=flipud(*A*)	将 *A* 矩阵上下翻转得到 *B* 矩阵
	B=flipdim(*A*, dim)	将 *A* 矩阵按给定的维数翻转得到 *B* 矩阵。dim=1 时，按行维翻转；dim=2 时，按列维翻转
	B=rot90(*A*,*k*)	将 *A* 矩阵逆时针旋转 $k\times90°$ 得到 *B* 矩阵，*k*=1 时可省略
矩阵提取函数	*B*=tril(*A*)	取 *A* 矩阵主对角线及以下元素得到 *B* 矩阵（即取矩阵 *A* 的下三角阵），其余位置元素补充 0
	B=tril(*A*,*k*)	取 *A* 矩阵第 *k* 条对角线及以下元素得到 *B* 矩阵（即取矩阵 *A* 的下三角阵），其余位置元素补充 0
	B=triu(*A*)	取 *A* 矩阵主对角线及以上元素得到 *B* 矩阵（即取矩阵 *A* 的上三角阵），其余位置元素补充 0
	B=triu(*A*,*k*)	取 *A* 矩阵第 *k* 条对角线及以上元素得到 *B* 矩阵（即取矩阵 *A* 的上三角阵），其余位置元素补充 0

注意：*k* 取 0 为主对角线；*k* 取正整数为主对角线上第 *k* 条对角线；*k* 取负整数为主对角线下第 *k* 条对角线。

【例 2-10】利用矩阵旋转函数将 *A* 矩阵转换为 *B* 矩阵。

```
A=[1 3 5;2 4 6;7 8 9]              %创建矩阵 A
A =
     1     3     5
     2     4     6
     7     8     9
B=fliplr(A)                        %将矩阵 A 左右翻转得到矩阵 B
B =
     5     3     1
     6     4     2
     9     8     7
B2=flipud(A)                       %将 A 矩阵上下翻转得到 B2 矩阵
B2 =
     7     8     9
     2     4     6
```

```
     1     3     5
B3=flipdim(A,1)          %将A矩阵按行维翻转得到B3矩阵
B3 =
     7     8     9
     2     4     6
     1     3     5
B4=flipdim(A,2)          %将A矩阵按列维翻转得到B4矩阵
B4 =
     5     3     1
     6     4     2
     9     8     7
B5=rot90(A,2)            %将A矩阵逆时针旋转2×90°得到B5矩阵
B5 =
     9     8     7
     6     4     2
     5     3     1
B6=rot90(A)              %将A矩阵逆时针旋转90°得到B6矩阵
B6 =
     5     6     9
     3     4     8
     1     2     7
```

【例 2-11】利用矩阵提取函数将 *A* 矩阵转换为 *B* 矩阵。

```
A=[1 3 5;2 4 6;7 8 9];   %创建同例2-10的矩阵A
B1=tril(A)               %取矩阵A的主对角线以下元素形成矩阵B1，其余位置补充0
B1 =
     1     0     0
     2     4     0
     7     8     9
B2=tril(A,-1);           %取矩阵A的主对角线以上第-1条对角线（即对角线以下第1
                         条对角线）以下的元素形成矩阵B2，其余位置补充0
B2 =
     0     0     0
     2     0     0
     7     8     0
triu(A)                  %取矩阵A的主对角线以上元素形成矩阵，其余位置补充0
ans =                    %如果没有指定输出变量，MATLAB默认将输出值赋给变量ans
1     3     5
0     4     6
0     0     9
```

```
triu(A,-1)          %取矩阵 A 的主对角线以上第-1 条对角线（即对角线以下第 1
                    条对角线）以上的元素形成矩阵，其余位置补充 0
ans =
     1     3     5
     2     4     6
     0     8     9
```

提示：ans 是 MATLAB 一个特殊的变量，用以输出程序计算的最近一次结果，自动输出未指定输出变量表达式的输出结果。下一次输出将覆盖上一次的输出结果。

2.1.5 矩阵的加、减、乘、除和乘方运算

矩阵的加、减、乘、除和乘方运算是最基本的矩阵运算，矩阵运算按照线性代数中基本的运算法则进行。MATLAB 系统提供了如表 2-3 所示的矩阵基本运算符及其等效的运算函数。

表 2-3　　基本矩阵运算符及其等效的运算函数

名称	运算符	等效的运算函数
加法	*A*+*B*	plus(*A*,*B*)
一元运算符加	+*A*	uplus(*A*)
减法	*A*-*B*	minus(*A*,*B*)
一元运算符减	-*A*	uminus(*A*)
乘法	*A***B*	mtimes(*A*,*B*)
矩阵左除	*A**B*	mldivide(*A*,*B*)
矩阵右除	*A*/*B*	mrdivide(*A*,*B*)
乘方	A^p	mpower(*A*,*p*)

说明：

- 矩阵的运算符及其相应的运算函数也就是算术运算符和相应的运算函数，可以同样应用于单个的数字；
- 一元运算符的加、减应用于矩阵运算中时，+*A* 表示取 *A*；-*S* 表示对矩阵 *A* 中的每个元素取负；
- 矩阵乘方 *A*^*p* 中，当 *p* 为正整数时，表示方阵 *A* 直接自乘 *p* 次；当 *p* 为负整数时，表示方阵 *A* 直接自乘 *p* 次后的逆；当 *p* 为零时，将给出和方阵 *A* 同维的单位阵。

注意：矩阵的运算必须要符合线性代数矩阵运算的要求：加减运算必须在具有相同行列的矩阵之间进行；只有当矩阵 *A* 的列数和矩阵 *B* 的行数相同时，才可进行矩阵 *A* 和 *B* 的乘法运算；乘方运算只有在矩阵为方阵时才有意义。如果矩阵的行和列不符合运算符的要求，MATLAB 将会自动给出错误信息。当一个矩阵和一个标量（1×1 的矩阵）进行运算时，其结果将是此标量和矩阵中的每一个元素"相加"、"相减"、"相乘"以及"相除"。在 MATLAB 中，矩阵左除和右除的含义不同。矩阵右除定义为：*B*\A=(*A*'/*B*')，用户只需掌握矩阵的左除即可。实际上矩阵的除法是矩阵乘法的逆运算，在实践中主要应用于解方程组。

【例 2-12】作矩阵 *A* 和 *B* 的加、减、乘、除和乘方运算。

```
A=[1 3 5;2 4 6;7 8 9];              %创建同例 2-10 的方阵 A
B=[1,2,3;4,5,6;7,8,9];
```

```
C1=A+B;
C1 =
     2     5     8
     6     9    12
    14    16    18
C2=A-B;
C2
C2 =
     0     1     2
    -2    -1     0
     0     0     0
C3=A*B;
C3
C3 =
    48    57    66
    60    72    84
   102   126   150
C4=A/B
Warning: Matrix is singular to working precision.
C4 =
   Inf   Inf   Inf
   Inf   Inf   Inf
   Inf   Inf   Inf
```

%此时 *B* 为奇异阵（singular），得出结果为无穷大（Inf）

```
C4=B/A
Warning: Matrix is close to singular or badly scaled.
```

Results may be inaccurate. RCOND = 1.096517e-017　　%当对矩阵作除法运算时，有可能因为误差设置的差别导致不精确的结果，此时，MATLAB 会自动给出警告信息

```
C4 =
         0    0.5000         0
   -0.5000    0.7917    0.4167
         0         0    1.0000
C5=A^2
C5 =
    42    55    68
    52    70    88
    86   125   164
```

说明：

- MATLAB 采用 IEEE（电气和电子工程师协会）规定的算法，即使 *A* 为奇异阵（即

A 的行列式值是 0)，运算也照样进行，但是此时 MATLAB 将给出警告信息："Warning: Matrix is singular to working precision."，求出的矩阵所有元素为无穷大（Inf）；

- 当矩阵 A 为病态阵（Badly Scaled）时，MATLAB 使用的算法产生的误差可能很大，MATLAB 系统也将给出警告信息："Warning: Matrix is badly scaled to working precision. Results may be inaccurate."；
- "Inf" 是 MATLAB 用来表示无穷大的专用变量。在 MATLAB 中，被零除或浮点溢出不会按错误处理，而是给出警告信息，同时用 "Inf" 作出标记。

2.1.6 矩阵函数

1. 基本的矩阵函数

线性代数中经常出现计算矩阵的行列式的值、求矩阵的秩以及特征值等运算，MATLAB 在这些方面有专门的运算函数。常用的矩阵运算函数如表 2-4 所示。

表 2-4　常用的矩阵运算函数

函数名称	功能和含义
Cond(A)	求矩阵 A 的条件数
det(A)	求方阵 A 的行列式值
dot(A,B)	求矩阵 A 和 B 的点积
eig(A)	求矩阵 A 的特征值和特征向量
Norm(A,1)	求矩阵 A 的 1-范数
Norm(A)或 norm(A,2)	求矩阵 A 的 2-范数
Norm(A, inf)	求矩阵 A 的无穷大-范数
Norm(A, 'fro')	求矩阵 A 的 F-范数
Rank(A)	求矩阵 A 的秩
Rcond(A)	求矩阵 A 的倒条件数
svd(A)	求矩阵 A 的奇异值分解
Trace(A)	求矩阵 A 的迹
Expm(A)	求矩阵 A 的指数 e^A
Expm1(A)	用 pade 法求矩阵 A 的指数 e^A
Expm2(A)	用 taylor 级数法求矩阵 A 的指数 e^A
Expm3(A)	用特征值和特征向量法求矩阵 A 的指数 e^A
Logm(A)	求矩阵 A 的对数
Sqrtm(A)	求矩阵 A 的平方根

注意：只有方阵才可计算行列式的值，即 det(A)的计算只有在 A 为方阵时才有意义。

expm2(A)采用泰勒级数法求矩阵的指数，此方法适用于任何矩阵，但精确度较差，运算速度较慢；expm3(A)采用矩阵的特征值和特征向量求矩阵的指数，此方法只有当独立特征向量的个数等于矩阵的秩时适用。

logm(A)和 sqrtm(A)计算矩阵的对数和平方根是指对矩阵 A 中每个元素求对数和平方根。

【例 2-13】用基本的矩阵函数计算矩阵 A 的各种值。

```
A=[3 3 5;2 4 6;7 8 9];        %创建方阵A
```

```
det(A)                %求方阵A的行列式值
ans =
    -24
eig(A)                %求特征值
ans =
    16.7503
     0.8793
    -1.6295
logm(A)               %求矩阵A的对数
ans =
    0.5432 + 0.8066i    0.7475 + 0.5526i    0.6902 - 0.6914i
    0.8584 + 1.4131i    0.7845 + 0.9681i    0.6967 - 1.2112i
     0.7502 - 1.5947i   1.1089 - 1.0926i    1.8504 + 1.3668i
sqrtm(A)              %求矩阵A的平方根
ans =
    1.2466 + 0.3278i    0.5192 + 0.2246i    1.0906 - 0.2809i
    0.2001 + 0.5742i    1.4228 + 0.3934i    1.3620 - 0.4921i
     1.6144 - 0.6480i   1.7430 - 0.4439i    2.3610 + 0.5554i
```

2. 矩阵分解函数

矩阵的分解是矩阵和数据分析的基础，在高等数学中占有十分重要的地位，MATLAB系统提供了大量的矩阵分解函数供用户选择使用，表 2-5 给出了常用的矩阵分解函数。

表 2-5　　常用的矩阵分解函数

函数名称	功能和含义
cdf2rdf(*V*, *D*)	将复数对角形式转化成实数块对角形式
Chol(*A*)	对矩阵 *A* 作 cholesky 分解
eig(*A*)	对矩阵 *A* 作特征值分解
Hess(*A*)	矩阵 *A* 的 hessenberg 形式
LU(*A*)	对矩阵 *A* 作 LU 分解
Null(*A*)	由奇异值分解得出的矩阵 *A* 的零空间的标准正交基
Orth(*A*)	矩阵 *A* 的行向量的标准正交基
Pinv(*A*)	求矩阵 *A* 的广义逆
qr(*A*)	对矩阵 *A* 进行 QR 正三角分解
qz(*A*)	对矩阵 *A* 进行 QZ 分解，用于广义特征值
rref(*A*)	将矩阵 *A* 转换为逐行递减的阶梯阵
rsf2csf(*V*,*D*)	将实数块对角形转化为复数对角形式
Schur(*A*)	矩阵 *A* 的 schur 分解
Subspace	计算由 *A*，*B* 张成的子空间夹角
svd(*A*)	对方阵 *A* 进行奇异值分解

【例 2-14】对矩阵 *X* 进行 QR 分解和 LU 分解。

```
X=[3 -1 2;1 2 -1; -2 1 4]    %输入矩阵X
X =
```

```
     3    -1     2
     1     2    -1
    -2     1     4
[L,U]=lu(X)                              %对矩阵 X 进行 LU 分解
L =
    1.0000         0         0
    0.3333    1.0000         0
   -0.6667    0.1429    1.0000
U =
    3.0000   -1.0000    2.0000
         0    2.3333   -1.6667
         0         0    5.5714
[Q,R]=qr(X)                              %对矩阵 X 进行 QR 分解
Q =
   -0.8018    0.1543    0.5774
   -0.2673   -0.9567   -0.1155
    0.5345   -0.2469    0.8083
R =
   -3.7417    0.8018    0.8018
         0   -2.3146    0.2777
         0         0    4.5033
Q*R
ans =
    3.0000   -1.0000    2.0000
    1.0000    2.0000   -1.0000
   -2.0000    1.0000    4.0000
```

2.2　数组函数和数组运算

MATLAB 同时提供矩阵（Matrix）和数组（Array）两种类型的数据。这两种数据类型的运算和定义既有相同之处，又有不同之处。本节内容将着重探讨数组的运算及其用法，同时给出与矩阵（Matrix）在运算及函数之间的差别。

2.2.1　数组和矩阵的区别

从外观形状和数据结构上看，二维数组和数学中的矩阵没有区别。但是矩阵作为一种变换或映射算子的体现，它和数组有着概念上的区别。通常，在非正式情况下，矩阵和数组的说法可以互相替换。确切地说，矩阵是数组的一种特例，即矩阵是二维的数值型数组，

表示一种线性变换的关系。定义在矩阵概念上的数学运算便构成线性代数这一课程。矩阵的运算遵循线性代数所规定的矩阵运算法则。

从运算的角度看，矩阵运算和数组运算有着显著的不同。在 MATLAB 中，矩阵运算和数组运算属于两种不同的运算：矩阵运算是从矩阵的整体出发，按照线性代数的运算规则进行，有着明确而严格的数学规则；而数组运算是从数组的单个元素出发，针对每个元素进行的运算，它是 MATLAB 软件所定义的规则，目的是为了数据管理的方便、操作简单、指令形式自然和执行计算有效。

另外，在 MATLAB 中针对矩阵和数组运算的算术运算符也有所不同，本书将在以后的部分遇到类似的情况时给予详细说明。

2.2.2 数组加、减、乘、除和乘方

数组的运算是对数组中每个元素进行的，这样就使大批数据的处理与标量的处理类似，可以大大简化使用和程序的编制，并且便于程序的阅读。

基本的数组运算符以及相应的运算函数如表 2-6 所示。

注意：对于加法和减法而言，矩阵运算和数组运算相同，并且要求两个执行加减运算的矩阵和数组的大小必须相同，否则将出现错误信息；对于乘法和除法而言，矩阵和数组的运算有着显著的不同：矩阵的乘除严格按照线性代数中规定的矩阵运算规则进行，而数组的运算则是针对数组中每个对应元素进行乘除运算。其运算符前必须加一个小黑点，表示按照数组的运算规则进行运算。

无论执行什么样的数组运算，所计算的结果数组总是与参与运算的数组同维。数组运算中所有的二元运算必须是同维的数组或者其中一个为标量，这一点与矩阵的运算不同。

表 2-6　基本数组运算符及其等效的运算函数

名称	运算符	相应的运算函数
加法	$A+B$	plus(A,B)
减法	$A-B$	minus(A,B)
数组乘法	$A.*B$	times(A,B)
数组左除	$A.\backslash B$	ldivide(A,B)
数组右除	$A./B$	rdivide(A,B)
乘方	$A.\^{}p$	power(A,p)

为了清晰地表达数组运算和矩阵运算的差别，表 2-7 中以对比的方式给出矩阵运算和数组运算命令以及功能和涵义。

注意：关于数组除的运算规则：①当参与除运算的两个数组同维时，运算为数组的相应元素相除，计算结果是与参与运算的数组同维的数组；②当参与运算的数组有一个是标量时，运算是标量和数组的每一个元素相除，计算结果是与参与运算的数组同维的数组；③右除与左除的关系为 $A./B=B.\backslash A$，其中 A 是被除数，B 是除数。

关于数组乘方的运算规则：①数组的标量乘方 $A.\^{}p$（即 A 为数组，p 为标量），运算为数组每个元素的 p 次方，计算结果是与数组 A 同维的数组；②标量的数组乘方 $p.\^{}A$，表示以 p 为底，分别以 A 的元素为指数求幂值，计算结果是与数组 A 同维的数组。

表 2-7 矩阵运算和数组运算的对照表

矩阵运算		数组运算	
命令形式	功能和含义	命令形式	功能和含义
A'	求矩阵 A 的共轭转置	A.'	求数组 A 的非共轭转置，相当于 conj(A')
		A=x	把标量 x 赋给 A 的每个元素
		x+A	标量 x 分别与数组 A 的元素之和
		x-A, A-x	标量 x 分别与数组 A 的元素之差
x×A	求标量 x 分别与矩阵 A 的各个元素之积	x.*A	标量 x 分别与数组 A 的元素之积
x×inv(A)	标量 x 与方阵 A 的逆阵求积	x./A, A.\x	标量 x 分别被数组 A 的元素除
A^n	方阵 A 自乘 n 次	A.^n	数组 A 的每个元素自乘 n 次
A^p	求方阵 A 的非整数乘方	A.^p	对数组 A 中每个元素分别求非整数幂
p^A	A 为方阵时，标量 p 的矩阵乘方	p.^A	以 p 为底，分别以 A 的元素为指数求幂值
A+B	矩阵相加	A+B	数组 A 和 B 的对应元素相加
A-B	矩阵相减	A-B	数组 A 和 B 的对应元素相减
A×B	内维相同的矩阵相乘	A.*B	数组 A 和 B 的对应元素相乘
A/B	矩阵 A 右除矩阵 B	A./B	A 的元素被 B 的对应元素除
expm(A)	A 的矩阵指数函数	exp(A)	以自然数 e 为底，分别以 A 的元素为指数，求幂
logm(A)	A 的矩阵对数函数	log(A)	对数组 A 的各元素求对数
sqrtm(A)	A 的矩阵平方根函数	Sqrt(A)	对数组 A 的各元素求平方根
funm(A,'fun')	A 的一般矩阵函数	f(A)	对数组 A 的各元素求函数值，f(.)表示表 2-6、表 2-7 所示的各数组函数
		A#B	数组 A，B 对应元素之间的关系运算，# 表示关系运算符
		A@B	数组 A，B 对应元素之间的逻辑运算，@ 表示逻辑运算符

【例 2-15】对 A 和 B 进行转置数组运算，并与矩阵运算的结果加以比较。

```
Clear              %清空内存中所有变量
A=zeros(3,3);      %为一个 3×3 的数组（矩阵）A 预先分配内存空间
A(:)=2:10;         %运用全元素赋值法对 A 的元素赋值
A=A*(1+2i)         %乘以一个复数产生复数数组（矩阵）
A =
    2.0000 + 4.0000i    5.0000 +10.0000i    8.0000 +16.0000i
    3.0000 + 6.0000i    6.0000 +12.0000i    9.0000 +18.0000i
    4.0000 + 8.0000i    7.0000 +14.0000i   10.0000 +20.0000i
A1=A.';            %对数组 A 进行数组转置，即求非共轭转置
A2=A';             %对矩阵 A 进行矩阵转置，即求共轭转置
A1
A1=
    2.0000 + 4.0000i    3.0000 + 6.0000i    4.0000 + 8.0000i
    5.0000 +10.0000i    6.0000 +12.0000i    7.0000 +14.0000i
```

```
        8.0000 +16.0000i    9.0000 +18.0000i    10.0000 +20.0000i
    A2
    A2 =
        2.0000 - 4.0000i    3.0000 - 6.0000i    4.0000 - 8.0000i
        5.0000 -10.0000i    6.0000 -12.0000i    7.0000 -14.0000i
        8.0000 -16.0000i    9.0000 -18.0000i    10.0000 -20.0000i
```

由例 2-15 可以看出，数组运算和矩阵运算有些运算符不同，其运算的结果也不同。

2.2.3 数组函数

MATLAB 系统本身带有强大的数学函数库，提供了两种类型的函数：一种是按照阵运算法则设计的称为矩阵函数；另一种按照数组运算法则进行设计的称为数组函数。标量可看作是单个的矩阵元素或数组元素，凡是可用于标量的数学函数都可用于矩阵和数组。下面介绍 MATLAB 基本的运算函数和用于数组的特定数组函数。

1. MATLAB 基本的运算函数

一般的数学计算使用 MATLAB 的基本运算函数可方便地实现。MATLAB 常用的数学函数如表 2-8 所示。三角函数和超越函数如表 2-9 所示。

表 2-8　MATLAB 常用的数学函数

函数名称	功能和含义
abs(x)	求数 x 的绝对值或求向量 x 的长度或求复数的模
angle(x)	求复数 x 的相角（Phase Angle）
conj(x)	求复数 z 的共轭复数
ceil(x)	天花板函数，即加入正整数至最近整数，即向正无穷方向取整
exp(x)	自然指数
fix(x)	无论正负数，舍去小数至最近整数
floor(x)	地板函数，即加入正整数至最近整数，即向负无穷方向取整
gcd(x,y)	求整数 x 和 y 的最大公因数
imag(x)	求复数 z 的虚部
lcm(x,y)	求整数 x 和 y 的最小公倍数
log(x)	求以 e 为底的对数，即求 x 的自然对数
log2(x)	求以 2 为底 x 的对数
log10(x)	求以 10 为底 x 的对数
Mod(x,y)	除法求余（与除数 y 同号）
log10(x)	求以 10 为底 x 的对数
Mod(x,y)	除法求余（与除数 y 同号）
Rem(x,y)	除法求余（与被除数 x 同号）
pow2(x)	求 2 的指数
rat(x)	将实数 x 化为分数表示
rats(x)	将实数 x 化为多项分数展开
real(x)	求复数 z 的实部
rem(x,y)	求 x 除以 y 的余数
round(x)	四舍五入至最近整数
sign(x)	符号函数（Signum Function） 当 x<0 时，sign(x)=-1；当 x=0 时，sign(x)=0；当 x>0 时，sign(x)=1
sqrt(x)	开平方根

表 2-9 常用三角函数和超越函数

函数名称	功能和含义	函数名称	功能和含义
sin(*x*)	求正弦函数	atan2(*x*, *y*)	求四象限的反正切函数
cos(*x*)	求余弦函数	asec(*x*)	求反正角函数
tan(*x*)	求正切函数	acsc(*x*)	求反余角函数
cot(*x*)	求余切函数	sinh(*x*)	求双曲正弦函数
sec(*x*)	求正角函数	cosh(*x*)	求双曲余弦函数
csc(*x*)	求余角函数	tanh(*x*)	求双曲正切函数
asin(*x*)	求反正弦函数	asinh(*x*)	求反双曲正弦函数
acos(*x*)	求反余弦函数	acosh(*x*)	求反双曲余弦函数
atan(*x*)	求反正切函数	atanh(*x*)	求反双曲正切函数

表 2-9 中的数学运算函数同样适用于数组。应用于数组时，对数组中的每个元素作函数运算。应用于矩阵时，按照矩阵运算法则进行运算。

2. 几个特殊的数组函数

MATLAB 提供有数组函数，用来对数组中的每个元素进行函数运算，运算结果和原数组维数一样的数组，MATLAB 提供了表 2-10 所示的一些特殊的数组函数。

表 2-10 一些特殊的数组函数

函数名称	功能和含义
besselj(*NU*, *Z*)	解第一类 Bessel 微分方程
bessely(*NU*,*Z*)	解第二类 Bessel 微分方程
beta(*Z*,*W*)	计算 beta 函数值
gamma(*X*)	计算 gamma 函数值
rat(*X*,tol)	计算 *X* 的有理近似（tol 表示容差）
erf(*X*)	计算 *X* 的误差函数
erfinv(*X*)	求逆误差函数
ellipke(*M*,tol)	求第一类、第二类全椭圆积分
ellipj(*U*,*M*)	jacobi 椭圆函数

提示：在表 2-10 所列的特殊的数组函数中，tol 表示容差，其余参数必须是和维数一样的数组或者是其中任意标量。关于函数调用和输出结果的含义，用户可参阅 MATLAB 的帮助文件，只需简单地输入“help 函数名”即可得到该函数的使用说明。

2.3 数据的输出

本节将讲解数据在 MATLAB 中的输出格式以及特殊变量和常数。

2.3.1 输出格式

在 MATLAB 中，程序或语句的执行结果都可以在屏幕上显示，同时可以赋值给指定

的变量。关于输出的格式问题，MATLAB 系统用 Format 命令来控制输出格式，并可用于在不同的输出之间转换。MATLAB 所有的计算采用双精度格式进行。Format 命令的各种选项和含义如表 2-11 所示。

表 2-11 Format 命令的格式

Format 命令	命令的含义
format	缺省输出格式与 short 相同
format short	5 位定点格式
format long	15 位定点格式
format short e	5 位浮点格式
format long e	15 位浮点格式
format short g	最佳 5 位定点格式或浮点格式
format hex	十六进制格式
format +	用+、- 符号和空格表示正、负数和零元素
format long g	最佳 15 位定点格式或浮点格式
format bank	用于美元和美分（金融）的固定格式，即小数点后两位
format rat	用有理分式表示
format compact	压缩额外的空行
format loose	显示变量之间插入空行

提示：Format 命令只影响结果的显示，不会影响计算和存储。

【例 2-16】Format 命令的应用。

```
a=3.141592657989323846;   %将π的值赋予变量a，精确到小数点后18位
format long               %按照15位定点格式输出
b1=a
b1 =                      %输出结果
3.14159265798932
format short
b2=a
b2 =
    3.1416
format short e
b3=a
b3 =
    3.1416e+000
format long e
b4=a
b4 =
    3.141592657989324e+000
format short g
b5=a
b5 =
    3.1416
```

```
format hex
b6=a
b6 =
    400921fb54db57c6
format +
b7=a
b7 =
+
format bank
b8=a
b8 =
    3.14
```

从本例可以看出，利用 Format 命令可以根据用户的要求成功地改变数据的输出格式，以满足用户的特殊需要。

2.3.2 特殊变量和常数

MATLAB 系统给出了一些常用的特殊变量和特殊常数，在程序的编制过程中极其有用，本节将详述如下。

1. 非数（NaN）

非数（Not-a-Number，NaN）的意思为“不是一个数值”，指无效的数值。按照 IEEE 规定，$\frac{0}{0}$、$\frac{\infty}{\infty}$、$0\times\infty$ 等运算均会产生非数，该非数在 MATLAB 中用 NaN 或 nan 表示。MATLAB 之所以引入非数的概念是因为：非数真实地记述了 $\frac{0}{0}$、$\frac{\infty}{\infty}$、$0\times\infty$ 运算的后果；非数的引用避免了可能因为执行了 $\frac{0}{0}$、$\frac{\infty}{\infty}$、$0\times\infty$ 运算而造成的程序执行的中断；在数据可视化中，可以使用非数来裁剪图形。

根据 IEEE 数学规范，非数具有以下性质：

- NaN 参与运算所得的结果也是 NaN，即 NaN 具有传递性；
- NaN 没有“大小”的概念，因此不能比较两个非数的大小；
- NaN 不能进行关系运算。

【例 2-17】演示非数的概念及其性质，并对非数元素进行寻访。

```
a=0/0                        %产生一个非数 a
Warning: Divide by zero.     %给出一个警告信息
a =
   NaN
b=log(0),c=inf-inf           %产生非数 b、c
Warning: Log of zero.
b =
```

```
    NaN
c =
NaN
sin(b)
ans =
NaN                          %可见非数具有传递性
a= =nan
ans =
0
b>c
ans =
    0                        %结果为假，可见非数无法进行关系运算
isnan(a)                     %询问 a 是非数吗
ans =
1                            %结果为真
```

2. “空”数组

“空”数组是 MATLAB 为了操作和描述的需要而专门设计的一种数组。在 MATLAB 5.x 以上版本中，凡是某维长度为 0 或若干维长度均为 0 的数组都是“空”数组。这与全零数组的概念不同。既有两维“空”数组，又有高维“空”数组。

MATLAB 之所以引入“空”数组的概念是因为：在没有“空”数组参与运算时，计算结果中的“空”可以解释，即“所得结果的含义”；运用“空”数组对其他非空数组赋值，可以改变数组的大小，但不能改变数组的维数。

可以用 Isempty 命令来判断一个数组是否是“空”数组，并且是惟一的判断方法。

注意：“空”数组和全零数组不同，二者是完全不同的两个概念，不要将“空”数组看作是“虚无”，它确实存在，可以用 which、who、whos 以及变量浏览器来验证它的存在；“空”数组在运算中同样不具备传递性。对于运算中出现“空”结果的解释要谨慎对待；有“空”数组参与的关系运算一般不能给出合理的结果，编程中用户要对产生“空”数组的情况特别注意，以免出现运行错误。

3. 特殊的变量

在 MATLAB 中有一些用于特定用途的特殊变量，较为常用的如下所述：

- ans 表示最近一次的输出结果；
- computer 计算机类型；
- eps 浮点运算的相对精度，即 MATLAB 中计算的相对精度；
- floaps 浮点运算的次数；
- *i*, *j* 复数中的虚数单位；
- inf 无穷大，即“*n*/0”或溢出的结果，MATLAB 定义了这个特殊的变量可以避免被 0 除造成的浮点溢出；
- pi 圆周率 π；
- realmax 计算机能够显示的最大浮点数；

- realmin 计算所能够显示的最小浮点数；
- version 字符串格式的 MATLAB 版本。

2.4 小　结

本章主要介绍了 MATLAB 最重要的基本计算单元——矩阵和数组，可以说本章是全书的基础。通过本章的学习，应该掌握如下内容：

（1）矩阵和数组的创建、使用和保存。

（2）熟练掌握矩阵运算和数组运算。

（3）学会如何使用矩阵运算函数和数组运算函数。

（4）注意区分矩阵和数组的差别，特别是运算符的差别。

（5）了解并初步掌握矩阵元素的标识和提取方法。

（6）了解 MATLAB 数据的输出格式以及常用的特殊变量和常数。

习　题

1．在 MATLAB 6.0 的工作空间中分别用直接输入法和矩阵编辑器建立如下两个矩阵，并将其保存。

$$A=\begin{bmatrix}1 & 3 & 5\\ 2 & 4 & 6\end{bmatrix}\qquad B=\begin{bmatrix}1 & 3 & 5\\ 2 & 4 & 6\\ 7 & 8 & 9\end{bmatrix}$$

2．分别对第 1 题中矩阵 A 和 B 作加、减、乘、除运算，同时运用数组运算法则进行运算，比较二者的计算结果有何异同。

3．利用矩阵生成函数建立一个对角线元素全部为 1 的 4 阶单位矩阵。

4．利用矩阵生成函数建立一个 4×4 的正态分布的随机矩阵，求该矩阵的特征值和特征向量，并练习用 diag 命令产生该矩阵的对角阵。

5．对第 1 题中的矩阵 B 求秩、行列式的值、条件数、平方根及对数。

6．用 format 的不同格式显示变量 pi 的值，并分析各个格式之间有何异同。

7．找出数组 $A=\begin{bmatrix}-4 & -2 & 0 & 2 & 4\\ -3 & -1 & 1 & 3 & 5\\ -1 & -3 & 5 & 7 & 9\\ -2 & -4 & 6 & 8 & 0\end{bmatrix}$ 中所有绝对值大于 3 的元素。

8．将如下数组 A 进行数组转置、对称交换和旋转操作，并将结果进行对照比较。

A =[-4 -1 2

-3 0 3

-2 1 4]

9．在 MATLAB 环境下，用下面三条指令创建二维数组 C，将输出怎样的结果。

a=2.7358;

b=33/79;

c=[1,2**a*+*i***b*,*b**sqrt(*a*);sin(pi/4),*a*+5**b*,3.5+*i*]

第3章　计算结果可视化

在一些试验、工程测量及科学计算中，经常会得到大量的离散数据点，称作采样点。在对这些数据利用之前，要对其进行分析和处理，因为对于大量的原始数据，很难直接从中找出内在的规律，而把这些数据用各种形式的图形表示出来，即把数据所反映的很多内在规律直观地展示在人们面前。因此，数据可视化是人们研究科学、认识世界不可缺少的重要手段。

MATLAB 在数据可视化方面提供了强大的功能，它可以把数据用二维、三维乃至四维图形表现出来。通过对图形的线型、立面、色彩、渲染、光线以及视角等属性的处理，将计算数据的特性表现得淋漓尽致。本章将通过对 MATLAB 在图形和数据可视化方面的功能进行介绍，学习如何利用 MATLAB 的图形功能来处理和解释数据所包含的内在规律。

3.1　MATLAB 的图形窗口

MATLAB 有一个用于图形输出的专用窗口，称为图形窗口（Figure Window）。通过该图形窗口，可以自由查看和设置有关图形输出及表达的参数，并获得高质量的图形文件。

3.1.1　创建与控制图形输出窗口

在执行一个绘图命令前，MATLAB 并没有打开图形窗口，只有在执行创建图形窗口命令后，系统才会自动创建一个图形窗口。若在命令执行前，已经存在了若干窗口，绘图命令会将图像输出到当前窗口，并将原来存在的该窗口的图像覆盖掉。

创建图形窗口的命令为 Figure，其有两种调用格式：

figure

figure(*n*)

直接在命令窗口中键入“figure”，将产生如图 3-1 所示的图形窗口，该窗口由菜单栏、工具栏和图形区三部分组成。和 MATLAB 5.3 版本相比，MATLAB 6.0 的图形窗口菜单栏多了【View】和【Insert】两项。

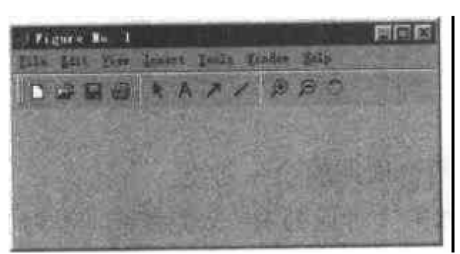


图 3-1　图形窗口

第一种命令格式生成的图形窗口名称是按照窗口创建的先后顺序依次命名的，如 Figure No.1，Figure No.2，…，Figure No.*n*，第二个命令将创建一个名为 Figure No.*n* 的新空白图形窗口，假如窗口 Figure No.*n* 已经存在，则将该窗口设置为当前窗口。

3.1.2 图形窗口的操作

图形窗口用于对各种图像进行操作，其操作方法与 MATLAB 工作窗口类似。关于图形窗口的独特命令及其参数的使用和具体设置方法将在本章以后的相关部分逐步介绍。

3.2 二维平面图形与坐标系

3.2.1 几个基本的绘图命令

1. 线性坐标曲线 plot

函数命令 plot 是 MATLAB 二维曲线绘图中最简单、最重要、使用最广泛的一个线性绘图函数。它可以生成线段、曲线和参数方程曲线的函数图形，对于不同的输入参数，该函数有不同的形式以实现不同的功能。同时，有许多其他特殊的绘图命令都以它为基础的。在这里我们将分别予以详细介绍。

- plot(*y*)

此命令格式中只有一个参数，以该参数的值为纵坐标，横坐标从 1 开始自动赋值为向量[1 2 3 4 ...]或其转置向量，向量的方向和长度与参数 *y* 相同。

【例 3-1】用命令 plot(*y*)绘曲线，其中 *y*=[5 3 4 9 0 2 3]。

在 MATLAB 的命令窗口中直接键入：

```
y=[5 3 4 9 0 2 3];
plot(y)
```

运行结果显示出如图 3-2 所示的曲线，其横坐标为向量[1 2 3 4 5 6 7]的元素值，纵坐标为向量 *y* 的元素值。

- 参数式 plot(*x*,*y*)

这是 plot 最常用的命令格式。参数 *x* 和 *y* 都是长度为 *n* 的向量，*x* 为横坐标向量，*y* 为纵坐标向量。这种调用可以用来生成参数方程的图形。

【例 3-2】绘制函数 *y*=cos(*x*)在两个周期内的图形。

在 MATLAB 的命令窗口中直接键入：

```
t=0:0.01:2*pi;
y=cos(t);
plot(t,y)
```

将绘出如图 3-3 所示的两个周期的余弦曲线。

注意：*x* 和 *y* 必须方向相同（行或列）、长度相等，否则 MATLAB 将提示错误信息。

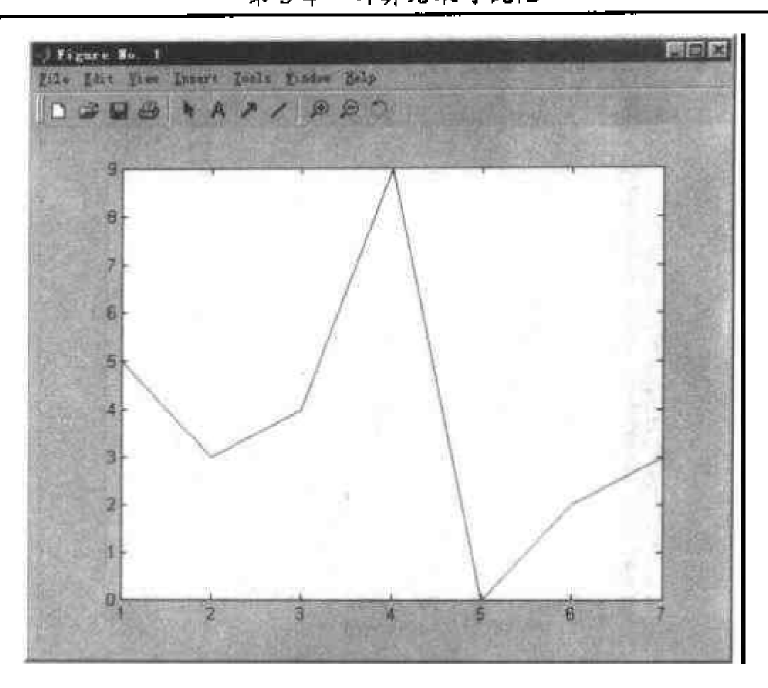


图 3-2　plot（y）绘出的图像

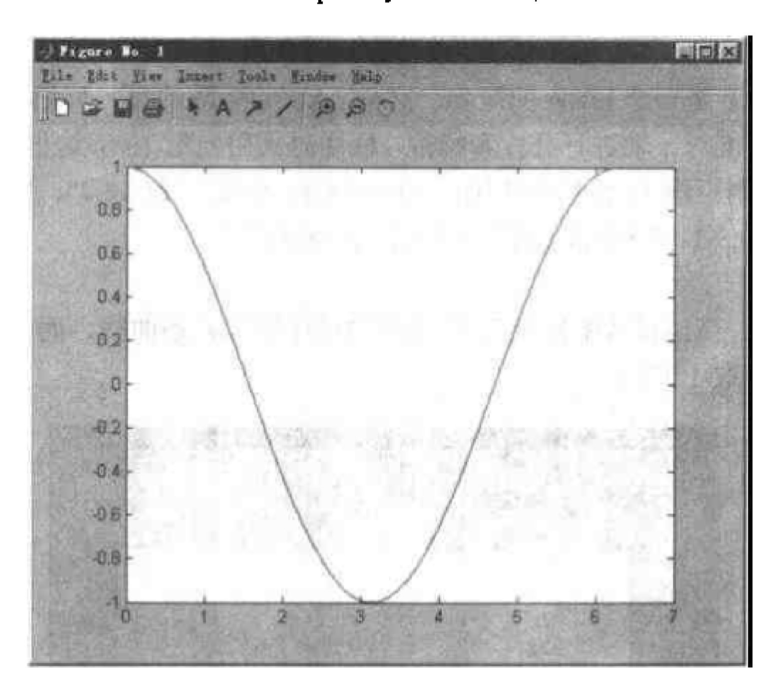


图 3-3　plot(*t*,*y*)绘出的图像

此种命令格式，还可以包括多个长度和向量 *x* 相等的列向量，此时将在图形窗口中绘制出多条曲线，如图 3-4 所示。

【例 3-3】在同一图形窗口中绘出正弦余弦函数的图形。

```
t=0:0.01:2*pi;
y=[sin(t);cos(t)];
plot(t,y)
```

绘出的曲线见图 3-4。

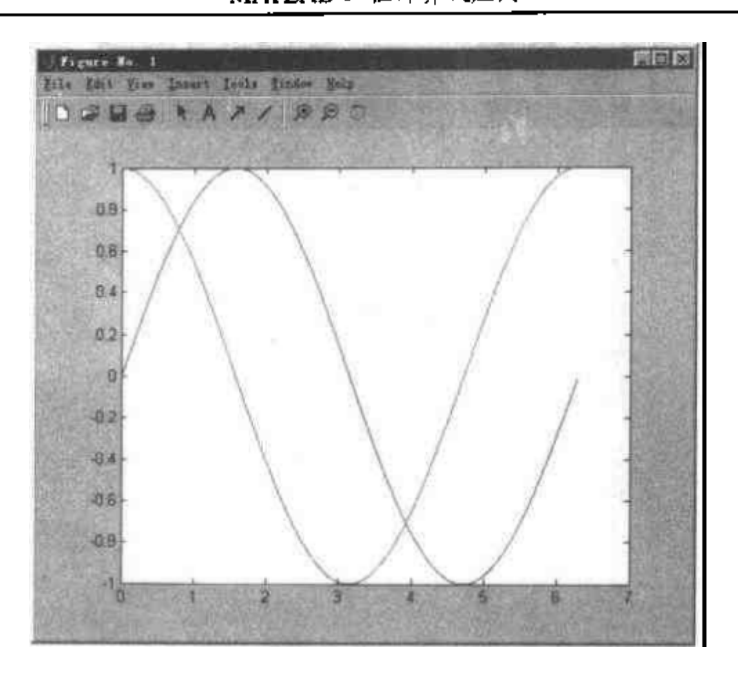


图 3-4　同一窗口中的正、余弦函数的图像

- plot(Y)

在 plot(Y)中，Y 是一个 $m \times n$ 的矩阵。MATLAB 为矩阵的每一列划出一条线，同时以矩阵的行向量为基准对 x 轴进行分度和标注，标注时采用向量 1:m，这里 m 是矩阵的行数。

【例 3-4】绘制矩阵 Y=[5 4 3 8 9 10；3 4 4 5 8 2；8 12 13 21 18 25；9 8 8 9 6 7]的图形。

```
y=[5 4 3 8 9 10;3 4 4 5 8 2 ;8 12 13 21 18 25;9 8 8 9 6 7];
plot(y)
```

如图 3-5 所示，MATLAB 按矩阵 Y 的列向量绘制出六条曲线，而 x 轴的取值则为矩阵 Y 的行向量，即取 1 2 3 4。

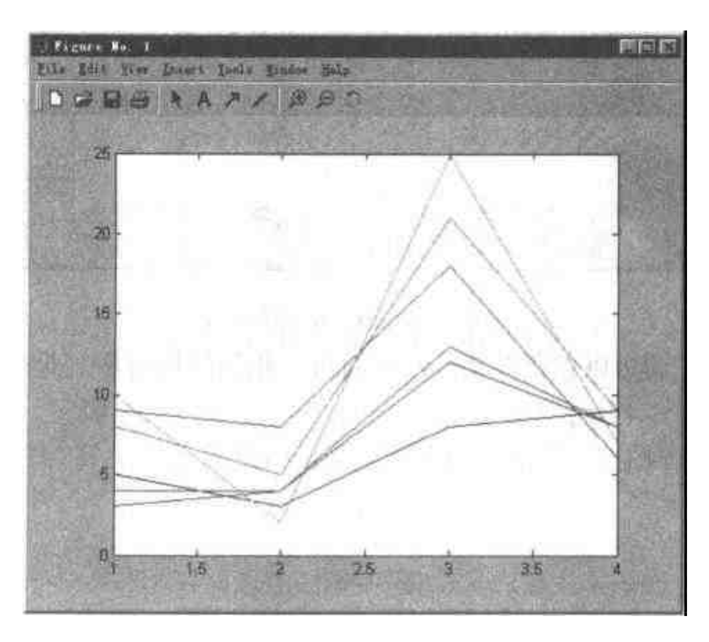


图 3-5　矩阵的图像

● 混合式 plot(*X*,*Y*)

在混合式的命令格式中，对于 *X* 和 *Y* 而言，可分下列几种情况：

◆ 如果 *X* 和 *Y* 都是向量，则长度必须相等；

◆ 如果 *X* 是向量，而 *Y* 是一个矩阵，*X* 的长度与矩阵 *Y* 的行数或列数相等，则它的作用是将向量 *X* 与矩阵 *Y* 的每列或每行的向量相对应作折（曲）线，当 *Y* 是方阵时，则将向量 *X* 与矩阵 *Y* 的列向量对应作图；

◆ 如果 *X* 是矩阵，*Y* 是向量，*Y* 的长度等于矩阵 *X* 的行数或列数，则将 *X* 的每列或每行的向量与 *Y* 相对应作图。当 *X* 是方阵时，则将 *X* 的各列与 *Y* 对应作图；

◆ 如果 *X* 和 *Y* 都是矩阵，且维数相同，则按列与列的对应方式来作图。

【例 3-5】绘制混合式的图形。

```
x=1:length(peaks);
plot(x,peaks)
```

执行结果如图 3-6 所示。

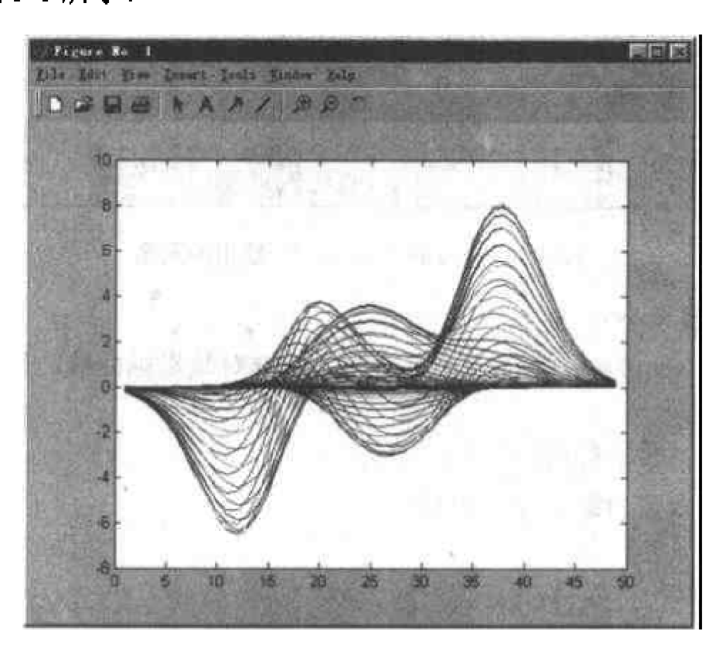


图 3-6　混合式 plot（*X*，*Y*）

● 复向量式 plot(*Z*)

当向量 *Z* 为一个复数向量时，MATLAB 将会忽略向量的虚部。也可以在调用时独立给出一个复参数，这时相当于两个指令的组合。例如，命令 plot(*Z*)和 plot(real(*Z*)，imag(*Z*) 是等效的（这里 *Z* 是一个复向量）。

● 综合调用方式：plot(*x*1,*y*1,*x*2,*y*2,…)

用这种形式也可以在同一窗口绘制多条曲线，而且每条曲线的横坐标可以不同，每一组向量也可以有不同的长度。

【例 3-6】在同一窗口中绘制多条曲线，并且坐标和长度都不同。

在 MATLAB 的命令窗口中直接键入：

```
t1=0:0.1:3*pi;
```

```
t2=0:0.1:6;
plot(t1,sin(t1),t2,sqrt(t2))
```

运行结果如图 3-7 所示，两条曲线的长度和坐标都不相同。

plot 命令是二维平面绘图中最重要、最常用的命令，读者可在具体的应用中加以体会、理解和掌握。

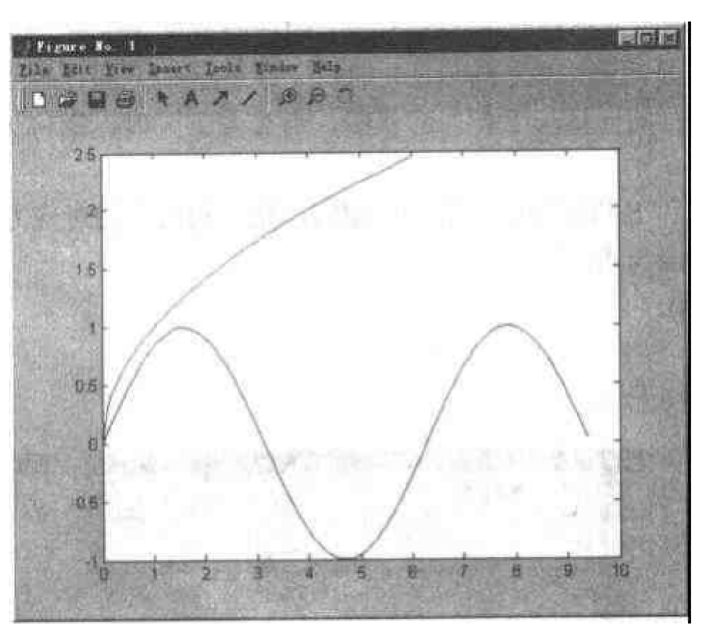


图 3-7 复向量式 plot（Z）绘出的图像

2. 对数坐标曲线命令

函数 semilogx、semilogy 和 loglog 用来绘制二维对数坐标曲线，这几个命令的用法和函数 plot 相同。

【例 3-7】绘制正弦函数的对数坐标曲线。

在 MATLAB 命令窗口键入如下内容：

```
t=0.1:0.1:3*pi;
y=sin(t);
semilogx(t,y)
grid on
```

最后一行命令 grid on 表示为图形窗口添加网格，其运行结果如图 3-8 所示。

注意：函数 semilogx 绘出的曲线，其横坐标为对数坐标；而函数 semilogy 绘制的曲线纵坐标为对数坐标；loglog 绘制的曲线，横纵坐标均为对数坐标。

3. 双 y 轴图形

用 plotyy 函数可以绘制左右均有 y 轴的图形，有以下几种常用的调用格式：

- plotyy(*x*1,*y*1,*x*2,*y*2)

此种调用格式可以在窗口中同时绘制两条曲线：(*x*1,*y*1)和(*x*2,*y*2)，曲线(*x*1,*y*1)用左侧的 *y* 轴，曲线(*x*2,*y*2)用右侧的 *y* 轴。

- plotyy(*x*1,*y*1,*x*2,*y*2, 'fun')

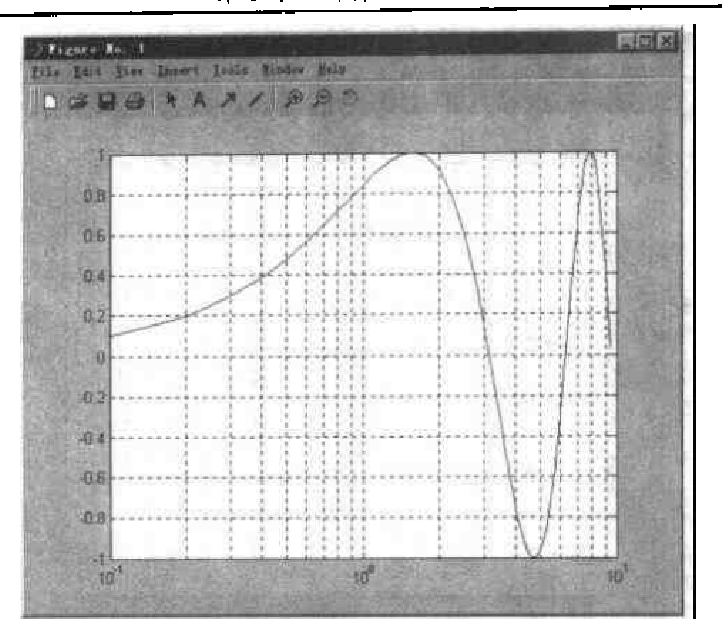


图 3-8 绘制对数坐标曲线

这种命令格式的使用与上面相同，只是多了'fun'一项。'fun'是字符串格式，用来指定绘图的函数名，如 plot、semilogx、semilogy 等。例如命令：

```
plotyy(x1,y1,x2,y2, 'semilogx')
```

就是用函数 semilogx 来绘制两条曲线，具体用法如例 3-8 所示。

【例 3-8】用 plotyy 函数绘制双 y 轴图形。

```
t1=0:0.1:3*pi;
t2=0:0.1:6;
y1=sin(t1);
y2=sqrt(t2);
plotyy(t1,y1,t2,y2,'semilogx')
grid on                    %为图形窗口添加网格
```

程序运行结果所绘的曲线如图 3-9 所示。

- plotyy(z1,y1,x2,y2, 'fun1', 'fun2')

同第二种命令格式相类似，只是用'fun1'和'fun2'指定不同的绘图函数分别绘制这两条曲线。其具体用法，将结合例 3-9 说明。

【例 3-9】在同一图形窗口中，用不同的绘图函数绘制同一函数曲线 $y=\sqrt{x}$ 的双 y 轴图形。

```
x=0:0.1:6;
y=sqrt(x);
plotyy(x,y,x,y,'semilogy','plot')
```

结果如图 3-10 所示。从图中可知，左侧的 y 轴为对数坐标，它与 semilogy 函数绘制的曲线对应；右侧的 y 轴为线性坐标，它与 plot 函数绘制的曲线对应。在彩色显示下，每条曲线和其对应的 y 轴颜色相同，而两个 y 轴的颜色互不相同。

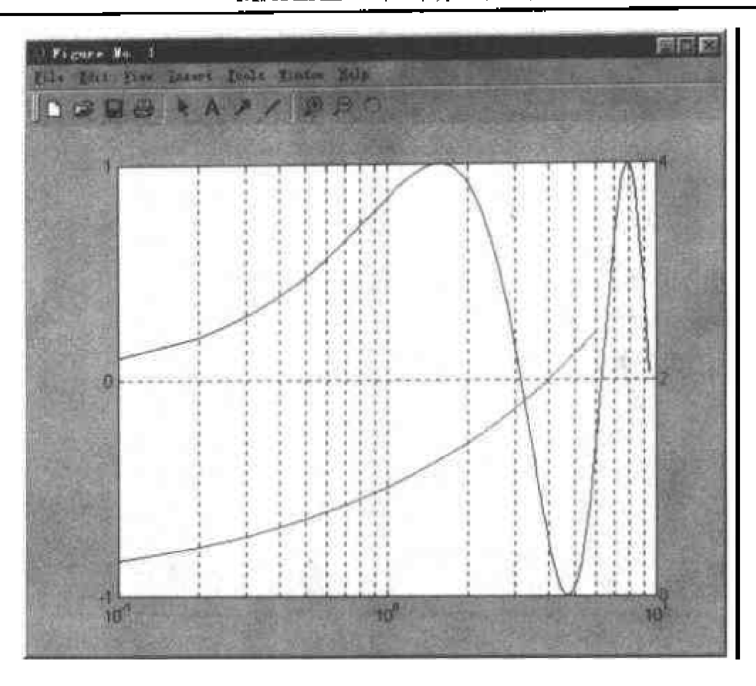


图 3-9　用指定函数名绘图

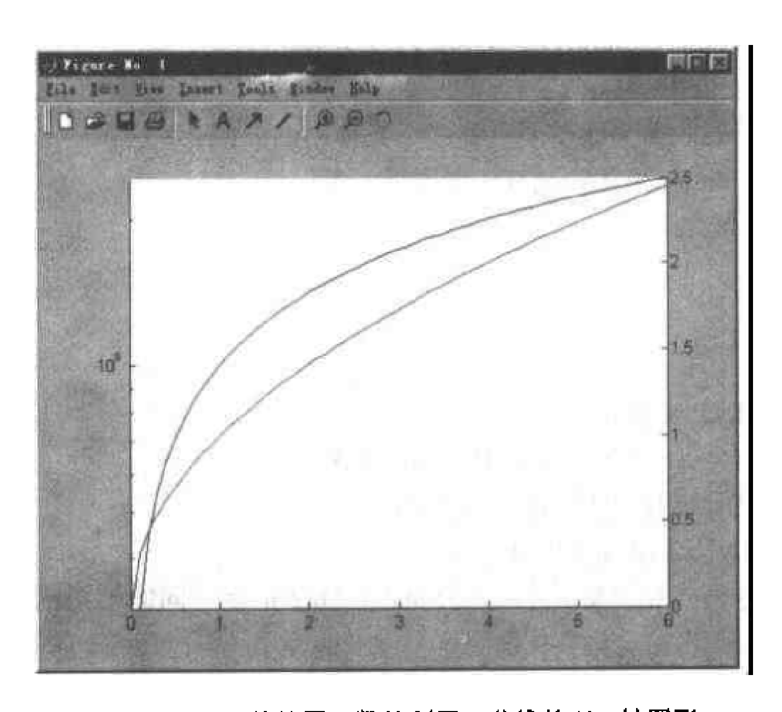


图 3-10　不同的绘图函数绘制同一曲线的双 y 轴图形

注意： plotyy 函数不能加入设置曲线线型、颜色及标出数据点的参数。

3.2.2　线型和颜色

前面介绍的 plot 函数有多种用法，在绘图时 MATLAB 自动安排作图的线型和线段的颜色以及线段顶点的标记。事实上，MATLAB 的 plot 函数可以设置和管理曲线的线段类型、顶点标记和线段颜色。

MATLAB 定义的常用线段类型、顶点标记和线段颜色的参数如表 3-1 所示。

表 3-1 MATLAB 常用的线段类型、线段颜色和顶点标记参数

颜色		线型		顶点标记			
符号	含义	符号	含义	符号	含义	符号	含义
b	蓝色	-	实线	.	实点标记	^	朝上三角符
g	绿色	:	虚线	o	圆圈标记	<	朝左三角符
r	红色	-.	点划线	×	叉字符标记	>	朝右三角符
c	青色	--	双划线	+	加号标记	p	五角星符
m	洋红			*	星号标记	h	六角形符
y	黄色			s	方块标记		
k	黑色			d	菱形标记		
w	白色			v	朝下三角符		

在 plot 函数中，最典型的调用方式是三元组参数，即：

plot(*x*,*y*,*s*)

其中，*s* 为类型说明参数，它是字符串，由表 3-1 中列出的符号组成。

使用过程中，应注意以下几点：

- *s* 字符串可以是三种类型的符号之一，也可以是线型与颜色和顶点标记与颜色的组合。

例如，字符串“—*r*”表示绘制红色的虚型；“:*yx*”表示绘制黄色点线，同时用符号“:*x*”标记数据点。

- 在 plot 函数指令中，如果没有 *s* 参数，plot 将使用缺省设置绘制曲线。MATLAB 缺省规定曲线一律用“实线”线型。不同曲线将按表 3-1 所给的前 7 种颜色次序着色。
- 通常在当前坐标系中绘图时，每调入一次绘图函数，如调用 plot 时，MATLAB 将擦掉坐标系中已有的图形对象。为了在一个坐标系中增加新的图形对象，可以用 MATLAB 的 hold on 命令达到这个目的。设置了 hold on 后，MATLAB 再生成新的图形时，保留当前坐标系中已存在的图形对象。此时，MATLAB 根据新图形的大小，可能会重新改变坐标系的比例。

此处将通过例 3-10 来说明用不同的线型和标注来绘制两条曲线所产生的效果。

【例 3-10】用不同的线型和标注来绘制两条曲线。

```
t1=0:0.1:2*pi;
t2=0:0.1:6;
y1=sin(t1);
y2=sqrt(t2);
plot(t1,y1,':hb',t2,y2,'--g')
```

运行结果如图 3-11 所示。

在图 3-11 中，正弦曲线用蓝色点线表示，其数值点用六角形绘制，而第二条函数曲线用绿色虚线表示。

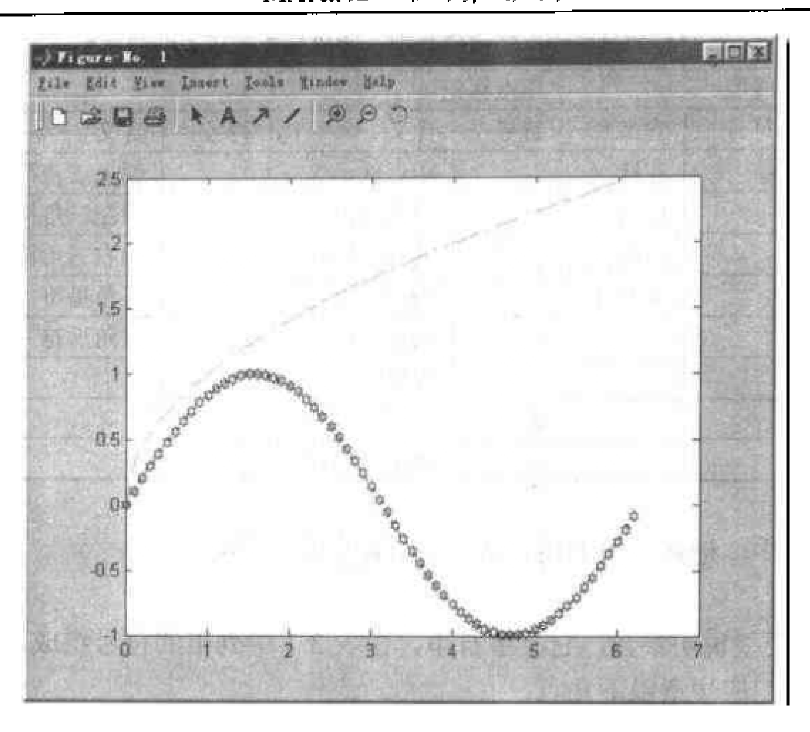


图 3-11 用不同的线型和标记绘制线形

3.2.3 二维数值函数曲线的专用命令 fplot

前面所介绍的绘图命令在绘制一个函数 $y=f(x)$的图形时，必须先定义自变量调用的一组取值点，再求出这组取值点的函数值，然后根据这两组数值确定的数据点绘制出所需的图形。而在实际应用中绘制函数二维曲线时一般并不清楚函数的具体情况，因而在确定自变量 x 的取值间隔时，一律用平均间隔，用这种方法绘制的的图形不够准确。

下面介绍绘制函数 $y=f(x)$图形的一个专用命令。该命令的特点在于它的绘图数据点是自适应产生的，即在函数曲线平坦处，它所取数据点比较稀疏，而在函数变化剧烈处，它将自动取较密的数据点。因此，对于那些导数变化较大的函数，用 fplot 函数绘出的曲线比等分取点所画出的曲线更加接近真实。fplot 函数命令的调用格式为：

[*X*,*Y*]=fplot(fun,lims,tol,n, 'linespec'，*p1*,*p2*...)

各参数含义如下：

- fun　函数名字符串，可以是一个由多个分量函数构成的函数行向量，分量函数可以是 MATLAB 的已有函数，也可以是用户自己定义的函数；
- lims　定义 x 的取值区间，lims=[xmin,xmax]；
- tol　相对误差，默认值为 2*e*-3，tol 越小，所绘制的曲线就越接近实际曲线的情况，但系统将为此占用很大资源；
- *n*　绘图的最少点数（*n*+1）；
- 'linespec'　线性设置；
- *p*1，*p*2，…　函数传递参数；
- *X*，*Y*　数组数据点坐标。

注意：上述命令的格式调用时，MATLAB 会把数据点坐标输入 *X*、*Y*，并没有用图形

显示出来，如需要显示图形，可以不加这两个参数；如果要使用 tol、n 或是'linespec'的默认参数，可以给函数传递一个空矩阵作为参数；在相同的数据下，自适应取点所绘的图形更加真实；自适应取点所用的时间较长。

下面将通过例 3-11 来对命令 plot 和 fplot 进行比较。

【例 3-11】fplot 和 plot 命令的比较。

在命令窗口中键入如下程序：

（1）创建函数 M 文件 funfplot。

```
function y=funfplot(x)
y=sin(1./tan(pi.*x));
```

（2）将用 fplot 命令求得的坐标点，按照对应大小，创建一个等间距坐标点赋值给矩阵 X，Y。

```
[X,Y]=fplot('funfplot',[-0.1,0.1],2e-4);
n=size(X);
x=-0.1:0.2/(n(1)+1):0.1;
y=funfplot(x);
```

（3）用 fplot 和 plot 命令作图，比较二者的不同：

```
plot(x,y)
figure        %打开一个新的图形窗口
plot(X,Y)     %该命令等价于 fplot('funfplot',[-0.1,0.1],2e-4);
```

运行结果如图 3-12、图 3-13 所示。

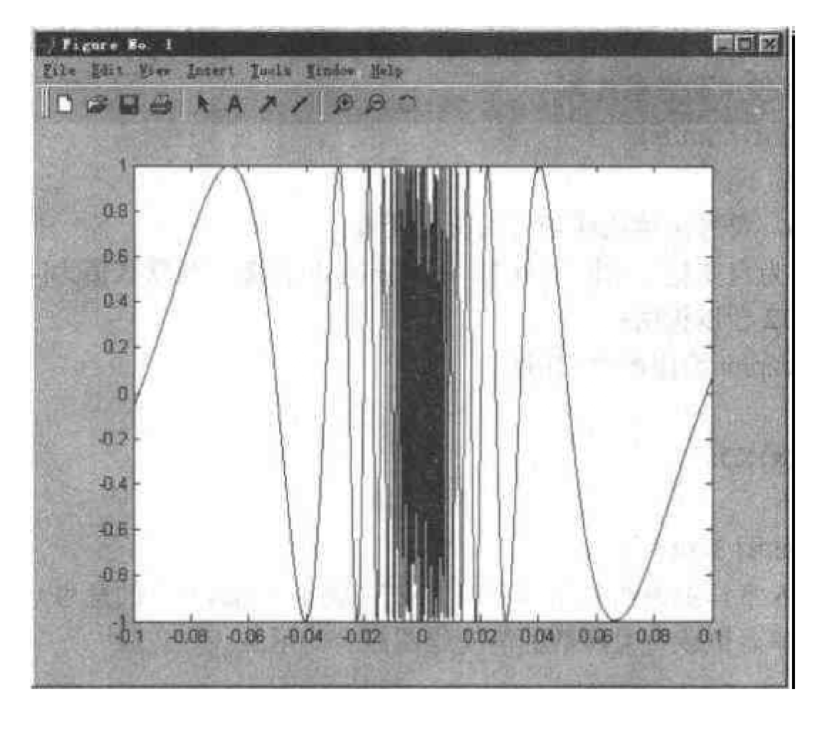


图 3-12 用 plot 命令绘图效果

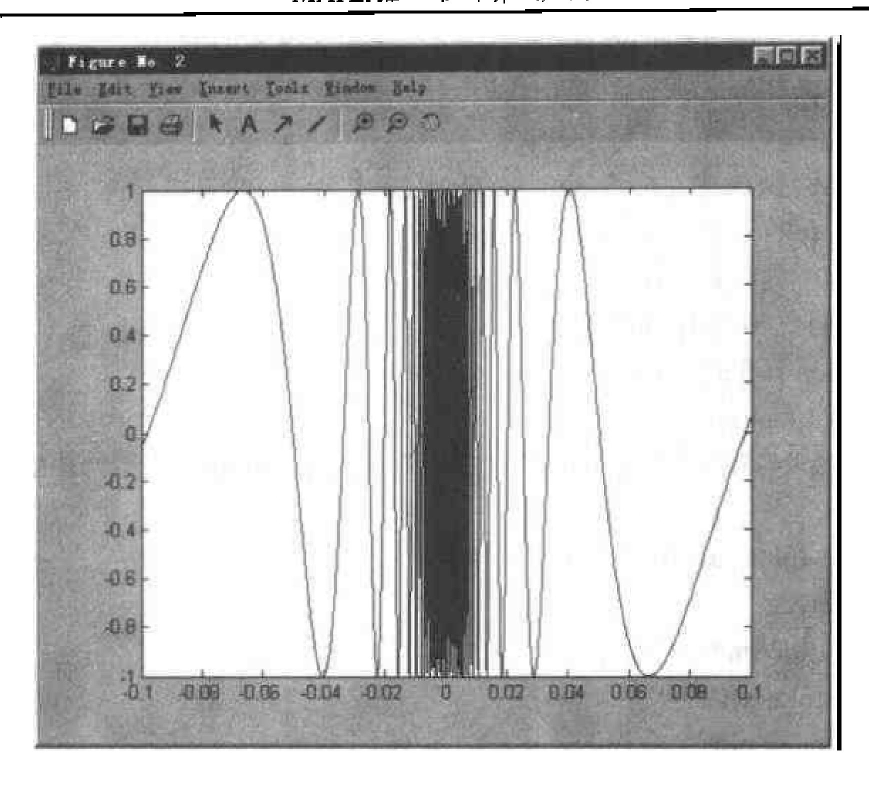


图 3-13 用 fplot 命令绘图效果

3.2.4 二维符号函数曲线的专用命令

为使用户更加方便地实现函数可视化，并完善符号函数（关于符号函数将在第 5 章详细说明）的图形功能，MATLAB 5.2 版以后，就有了绘制二维符号函数或字符串函数的专用命令 ezplot，其使用命令格式为：

ezplot(sym-fun，limits)

各参数含义如下：

- sym-fun 符号函数或代表它的符号变量；
- limits 为自变量 x 的取值范围，即 limits=[x1,x2]，其默认值为[-2pi,2pi]。

现通过例 3-12 予以说明：

【例 3-12】ezplot 绘图函数的用法。

```
syms x
f=(x^2)^(cos(x)^2);
ezplot(f)
```

其运行结果如图 3-14 所示。

本书将在第 5 章详细说明关于以字符“ez”（表示“easy to”的意思）开头，MATLAB 扩充的用于字符串函数或符号函数的简捷绘图命令的用法。

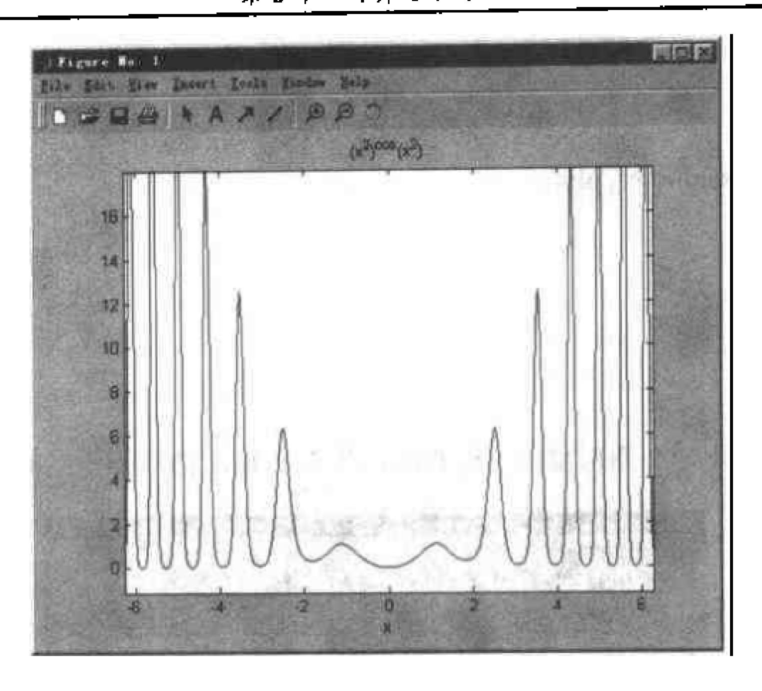


图 3-14　符号绘图函数 ezplot 绘制的图形曲线

3.2.5　图形窗口的分割

有时需要在一个图形窗口中显示几幅图，以便对几个函数进行直观的比较。由于每个绘图命令在绘制数据图像时都会将已有的图形覆盖掉，而用 hold 命令不能实现同时显示几个不同坐标尺寸下的图形，用 figure 命令再创窗口又很难同时比较由不同的数据绘得的图像。对于此类问题，MATLAB 创建关于实现在同一个窗口中同时显示多个图像的命令 subplot，其使用格式为：

subplot(*m*，*n*，*i*)

其含义为：把图形窗口分割为 m 行 n 列子窗口，然后选定第 i 个窗口为当前窗口。例如：subplot(2,2,4)意为把图形窗口分为 2 行 2 列共 4 个子窗口，选择第 2 行第 2 列（排序为 4）的子窗口为当前窗口进行操作。

【例 3-13】用 subplot 函数把两种不同的图形综合在一个图形窗口中。

```
subplot(2,2,1)
t=0.1:0.1:2*pi;
y=sin(t);
semilogx(t,y)
grid on

subplot(2,2,2)
t=0:0.1:4*pi;
y=sin(t);
plot(t,y)
```

```
subplot(2,2,3)
x=1:0.01:5;
y=exp(x);
plotyy(x,y,x,y,'semilogx','plot')

subplot(2,2,4)
x=1:0.1:10;
y=sqrt(x);
plot(x,y,':rd')
```

这一组命令在同一个图形窗口中绘制出如图 3-15 所示的 4 幅不同的图形。

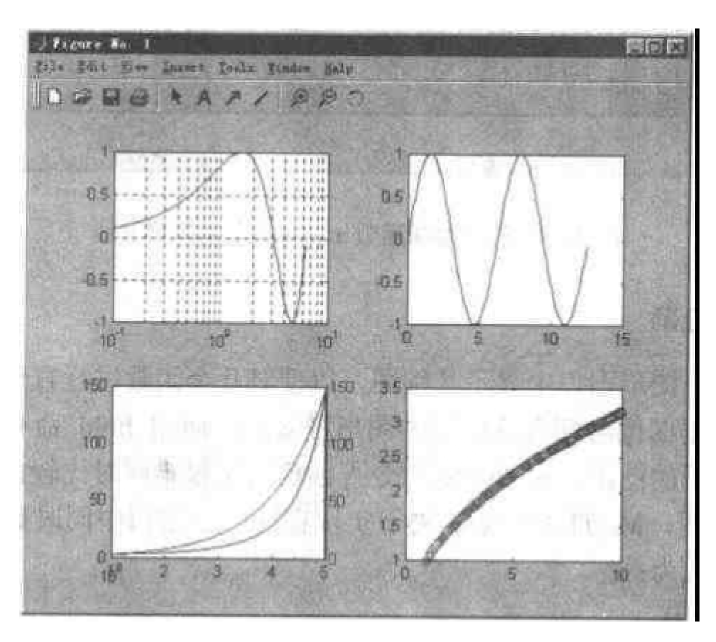


图 3-15 把一个窗口分为几个子窗口

注意：subplot 命令不仅用于二维图形，对三维图形一样适用。其本质是将 figure 窗口分为几个区域，再在每个区域内分别绘图。

- 如果坐标系存在，则将其设为当前坐标系。

命令 subplot('position',[left bottom width height])是在普通坐标系中创建一个新的坐标系。其中，各参量在 0 到 1 之间取值。

- 如果 subplot 命令所指定的区域与原有的区域重合（全部或部分），原区域将被删除。

subplot(111)是一个特殊的情况，它与 subplot(1,1,1)不同，该调用并不立刻创建坐标系，而是使下一条绘图命令在窗口中执行 clf 和 reset 命令（删除当前图形的所有子对象），然后在默认位置创建一个坐标系。这种调用没有返回值。

3.2.6 坐标系的调整

由前面的例子已经看到，MATLAB 的绘图函数可以根据要绘制的曲线数据范围自动地选择合适的坐标系，使曲线尽可能清晰地显示出来。因此，一般情况下用户不必自己选

择绘图坐标。如果觉得自动选择的坐标不太合适，则可以用手动的方式选择新的坐标系，在 MATLAB 中能实现此功能的命令就是 axis 函数。其调用格式为：

axis([xmin,xmax,ymin,ymax,zmin,zmax])

注意：坐标的最小值（xmin,ymin,zmin）必须小于相应的最大值（xmax,ymax,zmax），否则会出错。

【例 3-14】使用自动坐标系与用 axis 函数调整后的坐标系的比较。

```
subplot(2,1,1)
t=0:0.1:4*pi;
y=sin(t);
plot(t,y)

subplot(2,1,2)
t=0:0.1:4*pi;
y=sin(t);
plot(t,y)
axis([0,max(t),min(y),max(y)])
```

运行结果如图 3-16 所示。

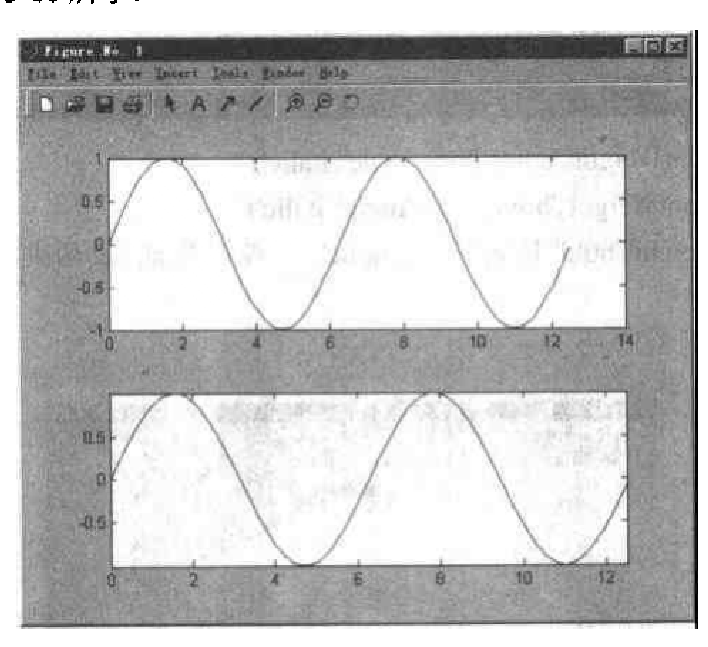


图 3-16 用 axis 命令调整前后的比较

3.3 三维绘图

MATLAB 6.0 具有强大的三维图形处理功能，包括三维数据显示、空间曲线、曲面、分块、填充以及曲面光顺着色、视点变换、旋转、隐藏等功能和操作。本节将介绍各种处

理命令，同时列出 MATLAB 6.0 的新增函数和功能。

3.3.1 基本的三维绘图命令

建立三维线条图的函数 plot3 和二维绘图函数 plot 相比，只多了第三维数据，其他与二维函数 plot 相同。其调用格式为：

plot3(*X1*,*Y1*,*Z1*,*s1*,*X2*,*Y2*,*Z2*,*s2*,…)或 plot3(*X*,*Y*,*Z*,*s*)

参数的含义如下：

- *Xn*、*Yn*、*Zn* 分别为第一到三维数据，是向量或矩阵，但必须尺寸相等；
- *s*、*s1*、*s2* 是可选的字符串，用来设置线型、颜色以及数据点标记等。

提示：*X*、*Y*、*Z* 是同维向量时，则绘制以向量 *X*、*Y*、*Z* 的元素为 *x*、*y*、*z* 坐标的三维曲线。

X、*Y*、*Z* 是同维矩阵时，则以 *X*、*Y*、*Z* 对应列元素为 *x*、*y*、*z* 坐标分别绘制曲线，曲线条数等于矩阵的列数。

绘线"四元组"（*X1*，*Y1*，*Z1*，*s1*），（*X2*，*Y2*，*Z2*，*s2*）的结构和作用与（*X*、*Y*、*Z*、*s*）相同。不同的是"四元组"之间没有约束关系。

三维线图指令 plot3 主要用来表现单参数的三维曲线。

【例 3-15】绘制一个三维螺旋线。

```
t=0:0.1:8*pi;
plot3(sin(t),cos(t),t)
title('绘制螺旋线')                                    %用命令 title 对图形主题进行标注
xlabel('sin(t)','FontWeight','bold','FontAngle','italic')
ylabel('cos(t)','FontWeight','bold','FontAngle','italic')
zlabel('t','FontWeight','bold','FontAngle','italic')   %命令 zlabel 用来指定 z 轴的数据名称
grid on
```

其执行结果如图 3-17 所示。

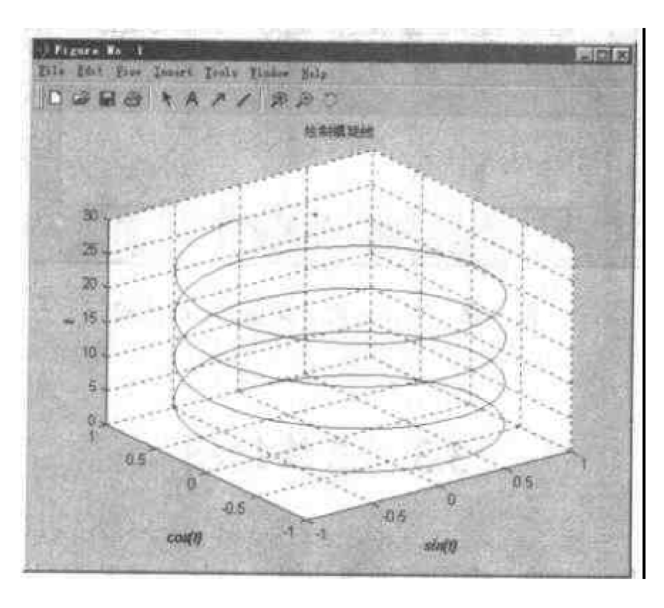


图 3-17 三维螺旋线

在图 3-17 中，如果把图中的 z 轴(t)去掉，即相当于从上往下看这幅图时，它是一个圆，这和 plot(sin(t),cos(t))绘制的曲线相同，由此可见，plot3 实际上就是二维函数 plot 在三维空间上的扩展。命令 zlabel 用来指定 z 轴的数据名称；命令 grid on 是在图底绘制三维网格；命令 title 是对图形主题的标注。

下面我们再看一个 x、y、z 是矩阵的例子。

【例 3-16】x、y、z 都是矩阵时，plot3 命令的使用。

```
[X,Y]=meshgrid(-pi:0.1:pi);
Z=sin(X)+cos(Y);
plot3(X,Y,Z)
```

其执行结果如图 3-18 所示。

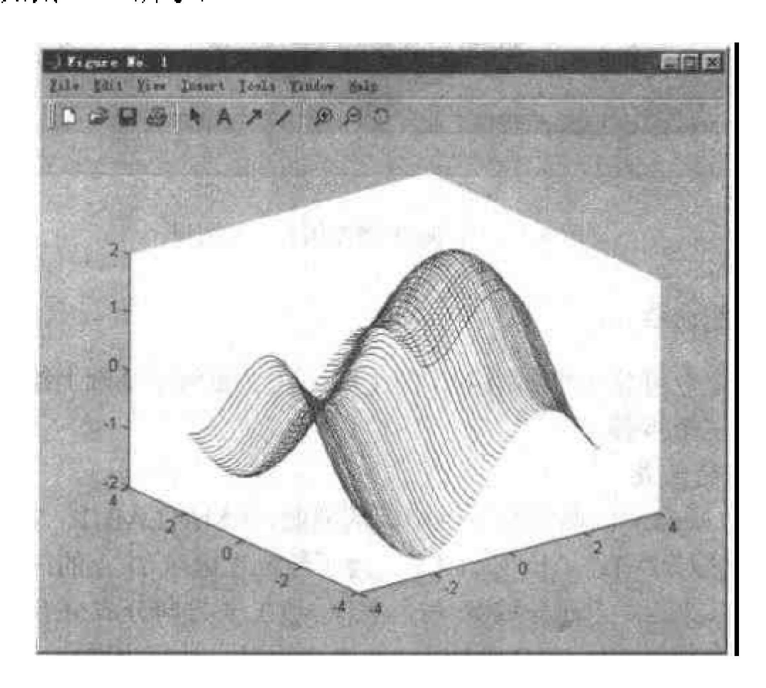


图 3-18　plot3 的参数为矩阵

3.3.2　线和面填色

在 MATLAB 中，函数 fill 和 patch 是图形的填充函数，它们的用法基本相同，下面仅以函数 patch 为例来说明这两个函数的使用。

Patch 函数的调用格式为：

绘制二维图形时　**patch(*X*，*Y*，*C*)**

绘制三维图形时　**patch(*X*，*Y*，*Z*，*C*)**

函数中参数 X、Y、Z 是向量或矩阵，C 用来指定颜色。

【例 3-17】用 patch 函数命令建立一个六边形，其填充颜色为红色。

```
patch([0,0.2,0.5,0.8,1,0.5,0],[1,0.3,0,0.3,1,1.8,1],'r')
```

其执行结果如图 3-19 所示。

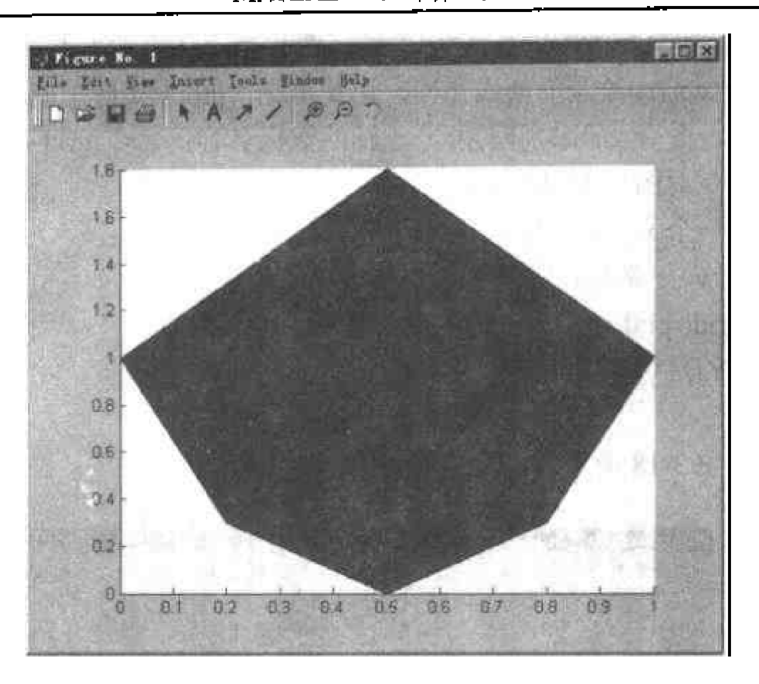


图 3-19 用 patch 命令填充后的图形

3.3.3 三维曲面绘图命令

三维曲面绘图命令可分为平面网格点的生成、在平面网格基础上绘制三维网格以及对三维表面进行处理三个步骤。

1. 平面网格点的生成

在数学上，函数 $z=f(x,y)$ 的图像是三维空间的曲面，在MATLAB中，总是假设函数 $z=f(x,y)$ 定义在一个矩形的区域 $D=[x_0,x_m]\times[y_0,y_n]$ 上。为了绘制在区域 D 上的三维曲面，MATLAB的方法是首先将 $[x_0,x_m]$ 在 x 方向分成 m 份，将 $[y_0,y_n]$ 在 y 方向分成 n 份，由各分划点分别作平行于坐标轴的直线，将区域 D 分成 $m\times n$ 个小矩形，计算出网格点的函数值。对于每个小矩形，在空间中决定出四个顶点（$xi,yi,f(xi,yi)$），连接四个顶点得到一个空间的四边形片。而所有这些四边形片连在一起构成函数 $z=f(x,y)$ 定义在区域 D 上的空间网格曲面。

在 MATLAB 中，用函数 meshgrid 命令来生成 x-y 平面上的小矩形顶点坐标值的矩阵。其调用形式为：

[*X*,*Y*]=meshgrid(*x*,*y*)

或**{*X*,*Y*}=meshgrid(*x*)**　　此种形式等价于**{*X*,*Y*}=meshgrid(*x*,*x*)**

格式中的参数含义如下：

- x 是区间 $[x_0,x_m]$ 上分划点组成的向量；
- y 是区间 $[y_0,y_n]$ 上分划点组成的向量；
- X，Y 为输出变量矩阵，矩阵 X 的行向量都是向量 x，矩阵 Y 的列向量都是向量 y。

这样，X 和 Y 的元素组（$X(i,j)$,$Y(i,j)$）恰好是区域 D 的第（i,j）网格顶点。例如，（$X(1,1),Y(1,1)$）对应于 $(x0,y0)$ 点，而（$X(m+1,n+1),Y(m+1,n+1)$）对应于（xm,xn）点。也就是说，函数 meshgrid 将有两个向量决定的区域转换为对应的网格点矩阵。

在计算网格点处的函数值时，由于矩阵 X 和 Y 的对应元素恰好组成某个网格点，因

此利用 MATLAB 的矩阵运算能力，可以很容易地求出由所有网格点上的函数值组成的矩阵。下面通过例子来具体说明其用法。

【例 3-18】数学函数，$z = x \times e^{(-x^2-y^2)}$ 其定义区域[-2，2]×[-2，2]。在生成网格后，计算网格点上的函数值。

```
[X,Y] = meshgrid(-2:2:2, -2:2:2);
[X,Y];                                  %将划分结果输出至矩阵X，Y
ans =
      -2     0     2    -2    -2    -2
      -2     0     2     0     0     0
      -2     0     2     2     2     2
Z = X .* exp(-X.^2 - Y.^2);             %计算网格点上的函数值赋予变量Z
Z =
      -0.0007        0     0.0007
      -0.0366        0     0.0366
      -0.0007        0     0.0007
```

2、三维网格命令 mesh

在得到了网格点上的函数值矩阵后，可以利用 MATLAB 中函数 mesh 来生成函数的网格曲面，即各网格线段组成的曲面。Mesh 函数的调用格式如下。

- esh(*X*，*Y*，*Z*，*C*)　这是最一般的调用形式。*X*、*Y*、*Z*、*C* 是同维数的矩阵，*X*、*Y*、*Z* 对应确定空间上的网格点，*C* 为颜色矩阵。也就是说，网格曲面的顶点对应于空间的顶点（*X*（*i,j*）,*Y*(*i,j*)，*Z*(*i,j*)），而网格曲面的网格线颜色由 *C* 的值根据当前的色谱来着色。此种调用形式还可以用来生成参数曲面片。
- mesh(*X*，*Y*，*Z*)　调用形式如同 esh(*X*，*Y*，*Z*，*C*)中 C=Z 的情况。
- mesh(*x*，*y*，*Z*，*C*)　其中，*x* 和 y 是向量，Z 和 C 是同维数的矩阵，并且向量 *x* 的长度等于矩阵 Z 的列数，而向量 y 的长度等于矩阵 Z 的行数。即 length(*x*) = *n*，length(*y*) = *m*，这里[*m*,*n*] = size(*Z*)。此时网格曲面的网格顶点是（*x*(*j*)，*y*(*i*)，*Z*(*i*,*j*)），网格线的颜色由矩阵 C 决定。
- mesh(*x*，*y*，*Z*)　调用形式如同 mesh(*x*，*y*，*Z*，*C*)中 C=Z 的简单调用形式。
- mesh(*Z*，*C*)　Z 和 C 都是 *m*×*n* 的矩阵，等价于 mesh(*x*，*y*，*Z*，*C*)，只是此时向量 *x* = 1:*n*，向量 *y* = 1:*m*。
- mesh(Z)　调用形式如同 mesh(*Z*，*C*)中 *C*=*Z* 的简单调用形式；
- mesh(...,'PropertyName',PropertyValue,...)　此种调用形式是给函数 mesh 设置曲面属性，具体用法同前面。

通过例 3-19 来说明函数 mesh 的用法。

【例 3-19】用 mesh 命令绘制例 3-18 中的网格曲面。

```
[X,Y] = meshgrid(-2:.2:2, -2:.2:2);
Z = X .* exp(-X.^2 - Y.^2);
mesh(Z)
```

生成的图形如图 3-20 所示。

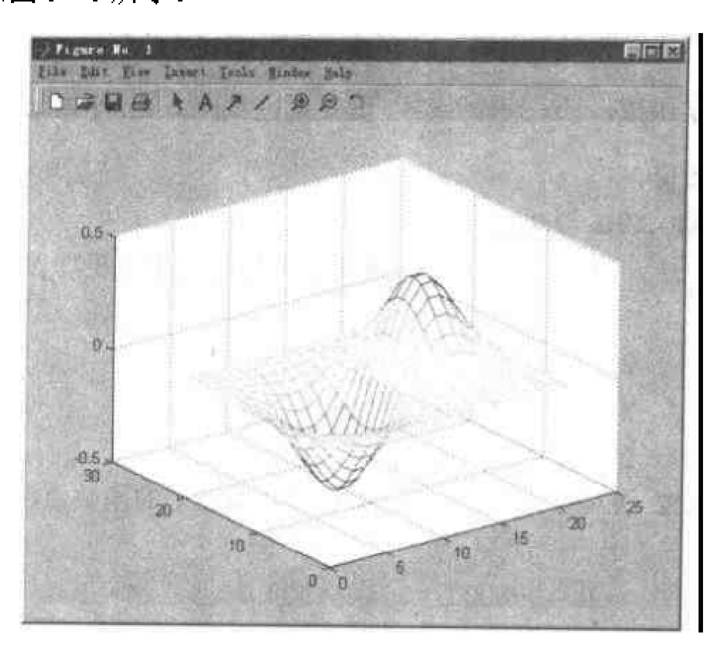


图 3-20 mesh 函数命令生成的网格曲面

与 mesh 相关的另外两个函数是 meshc 和 meshz，它们的调用形式与 mesh 相同。其区别在以下方面：

- meshc 除生成网格曲面外，还在 *x-y* 平面上生成曲面的等高线图形，如图 3-21 中图 a 函数 meshc 的绘图所示；
- meshz 除生成与 mesh 相同的网格曲面外，还在曲面下面加上一个长方体的台柱，其图形更加美观，如图 3-21 中图 b 函数 meshz 的绘图所示。

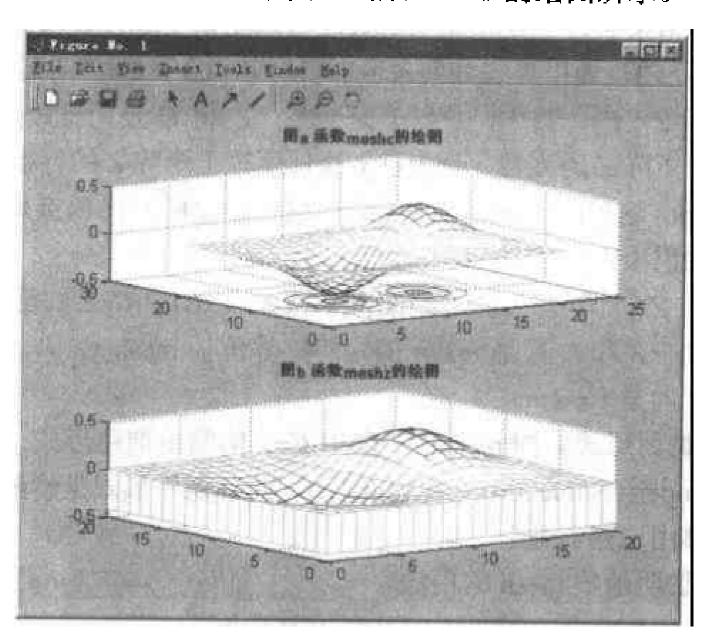


图 3-21 meshc 和 meshz 生成的图形

【例 3-20】演示函数 meshc 和 meshz 的用法，注意比较它们的不同。

```
[X,Y] = meshgrid(-2:.2:2, -2:.2:2);
Z = X .* exp(-X.^2 - Y.^2);
Subplot(2,1,1)
meshc(Z)
subplot(2,1,2)
meshz(Z)
```

3. 三维表面命令 surf

实曲面是对网格曲面的网格块区域进行着色的结果。在 MATLAB 中，函数 surf 可实现对网格曲面进行着色，将网格曲面转化为实曲面。surf 命令的调用格式与 mesh 相同，这里不再重复，仅仅给出例子加以说明。

函数 surf 的曲面生成过程与 mesh 相似，但着色机理与 mesh 不同。mesh 命令仅对网格线着色，而 surf 是对网格片着色，网格线用黑色标出（默认）。一般情况下，surf 用默认的着色方式对曲面片着色，还可以用 MATLAB 的函数 shading 来改变着色方式。

【例 3-21】利用三维网格表面命令 surf 绘制图形。

```
z=peaks;    %绘制山峰的图像，将函数值赋予变量 z
surf(z)     %对山峰的图像进行着色处理
shading interp
```

其运行结果如图 3-22 所示。

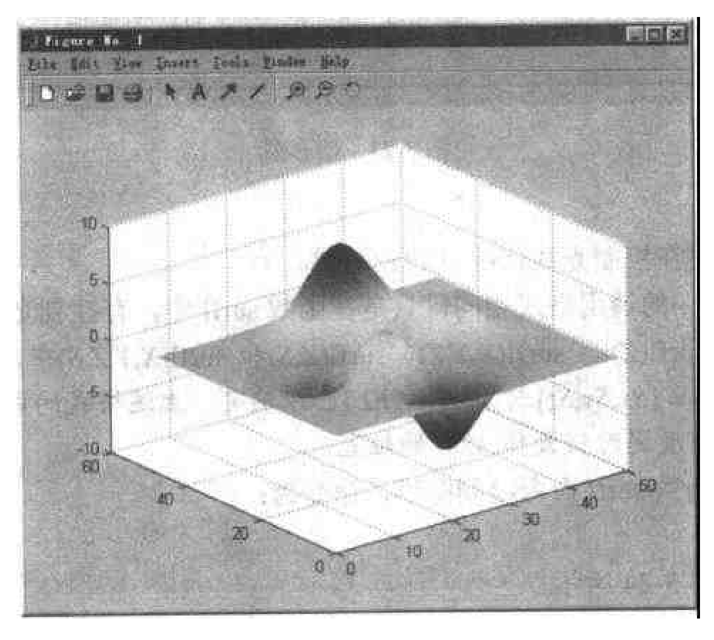


图 3-22　三维表面图

3.3.4　基本三维绘图命令的几个改进命令

在基本三维绘图命令的基础上，MATLAB 6.0 提供了一些对其功能进行改进和加强的命令。

1. pcolor 命令

pcolor 命令用于绘制伪彩图，其调用命令格式为：

- pcolor(*Z*) 以矩阵 Z 的下标为横坐标绘制伪彩图；
- pcolor(*x*，*y*，*z*) 以向量 *x*、*y* 为横坐标绘制伪彩图。

【例 3-22】用函数 pcolor 绘制伪彩图，命令为：

```
pcolor(peaks)
```

其执行结果如图 3-23 所示。

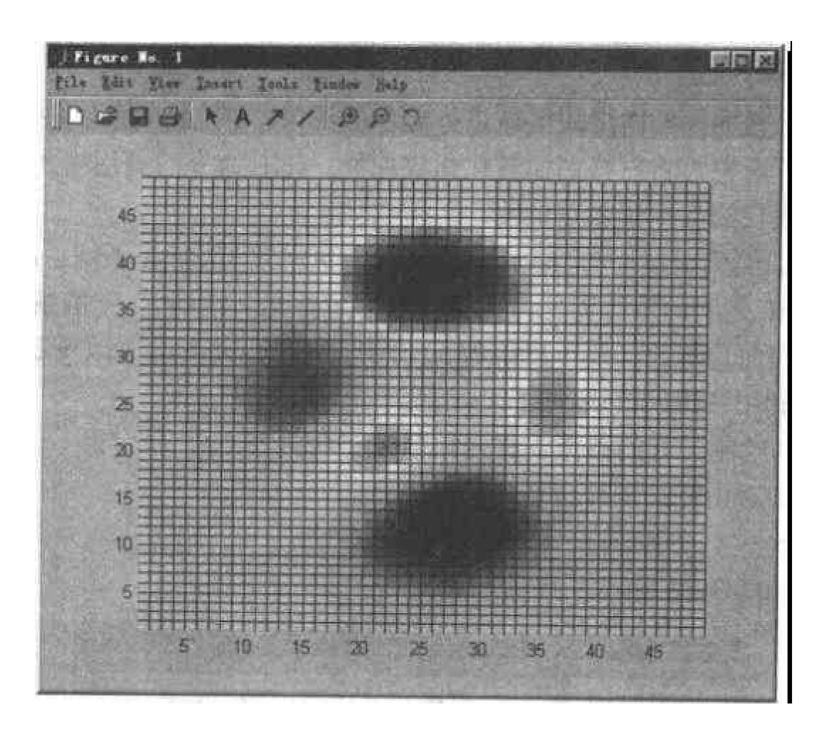


图 3-23 伪彩图

2. surfl 命令

此命令用于绘制在控制光线情况下的表面图。

surfl(...)与 surf(...)的调用格式相同，只是在函数 surfl 中，有控制光线视角的功能。

注意：在格式 surfl(Z)、 surfl(X,Y,Z)、surfl(Z,S)和 surfl(X,Y,Z,S)中，S 是光源位置。如果 S 具体确定，即 S = [Sx,Sy,Sz]或 S = [AZ,EL]已经确定，上述格式的调用相同；默认情况下，光源在从视线角度逆时针旋转 45° 的位置。

【例 3-23】用函数 surfl 命令绘制图形，命令为：

```
surfl(peaks)
```

其执行结果如图 3-24 所示。

3. waterfall 命令

此命令用于绘制类似瀑布流水形状的网格图，其使用格式和 mesh 命令基本相同，只是不画纵向的线条，因而产生类似“瀑布”的效果。

【例 3-24】绘出三维高斯分布的瀑布图。

```
waterfall(peaks)
axis tight
```

其执行结果如图 3-25 所示。

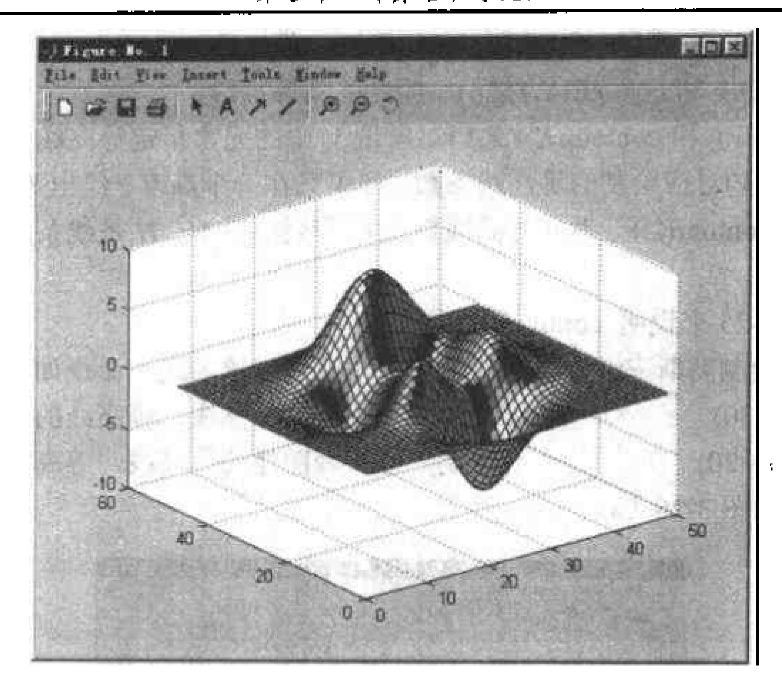


图 3-24 光线受控图

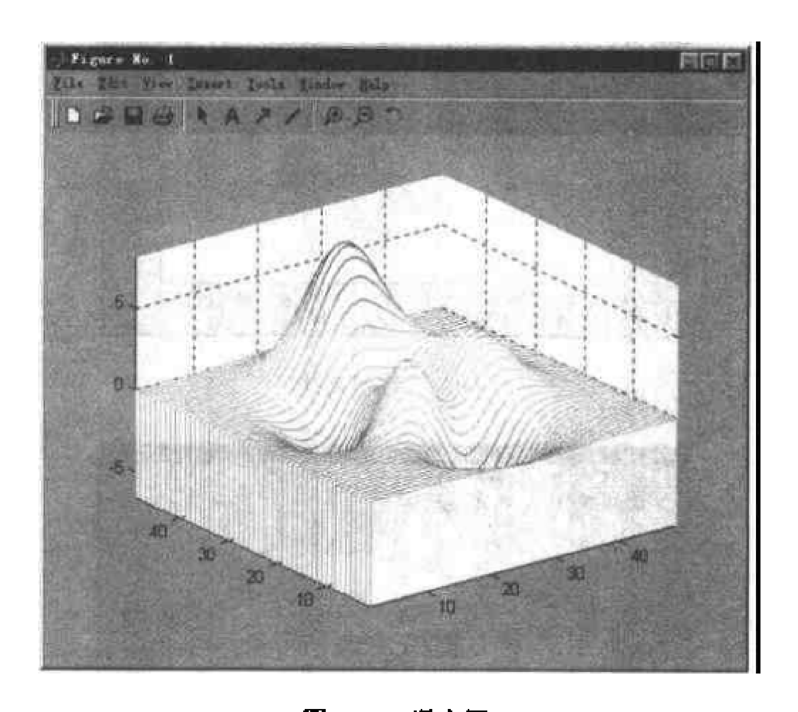


图 3-25 瀑布图

3.3.5 等高线图形的绘制、标注和填充

MATLAB 支持二维和三维的等高线图形，函数 contour 和 contour3 可绘制出二维和三维图形的等高线。用户可以指定等高线的条数、坐标系的比例及某值上的等高线。contour 和 contour3 的调用方式相同，只是 contour3 要绘制相应的 z 轴。下面以 contour 为例，说明它们的调用方式：

- contour(Z)　直接绘制矩阵 Z 的等高线；

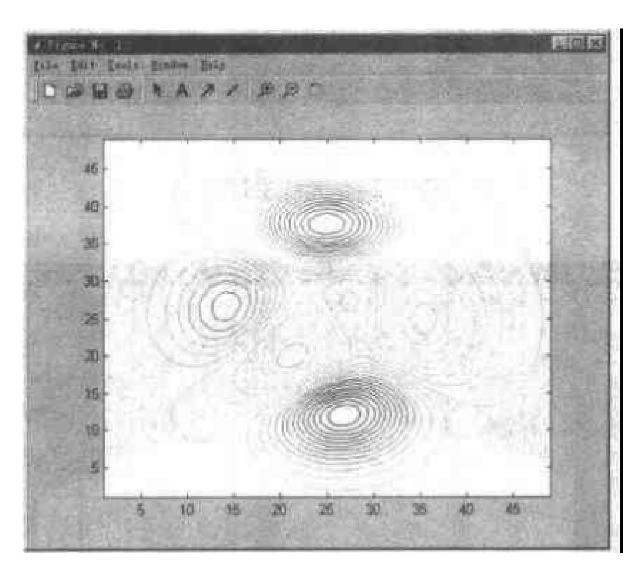
- contour(X,Y,Z) 用 X 和 Y 指定等高线的 x、y 坐标；
- contour(Z,n) 和 contour(X,Y,Z,n) 绘制 n 条等高线；
- contour(Z,V) 和 contour(X,Y,Z,V) 向量 V 中的元素指定等高线的位置，该向量长度 length(V)对应绘制的等高线条数。如果只在一个高度 z 绘出等高线，则 V=[z,z]；
- [C,H] = contour(...) 返回等高线矩阵 C 和列向量 H，H 是线条对象或补片对象的句柄。

下面通过例 3-25 来说明 contour 的用法。

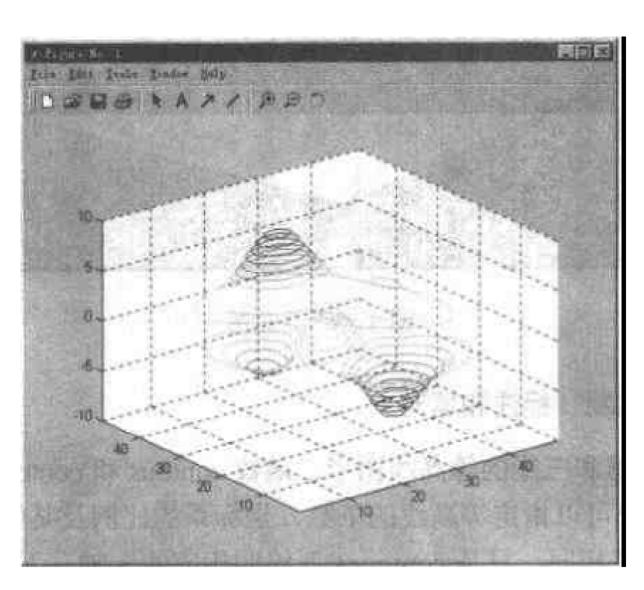
【例 3-25】绘制高斯三维分布曲面的等高线及二维等高线。命令如下：

```
contour(peaks,30)            %绘制二维山峰的等高线，等高线的条数为 30
contour3(peaks,20)           %绘制三维山峰图像的等高线，等高线的条数为 20
```

其执行结果如图 3-26（a）、（b）所示。



（a）二维等高线



（b）三维等高线

图 3-26 等高线

绘制好等高线后，可以用函数 clabel 标注高度值，其调用格式为：

- clabel(*C*)　将绘制的等高线全部自动标注，*C* 为等高线矩阵；
- clabel(*C*,*V*)　自动标注由向量 *V* 确定的若干条等高线的高度值，此处的向量 *V* 必须是前面 contour 命令中 *V* 的子集；
- clabel(C, 'manual')　手工标注等高线高度。

【例 3-26】用函数 clabel 标注山峰的等高线。

```
[C,H]=contour(peaks,20);
clabel(C,H)
```

其执行结果如图 3-27 所示。

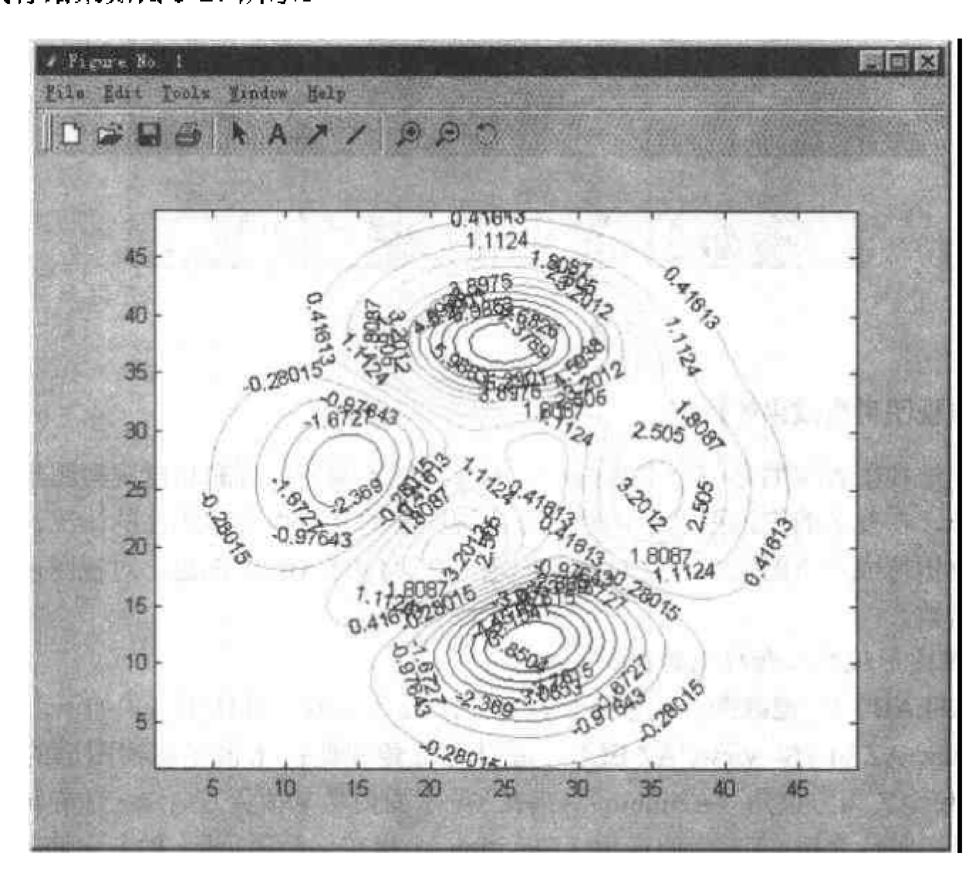


图 3-27　等高线的标注

在 MATLAB 中，还可以用 contourf 函数绘制填充的二维等高线，该函数的基本用法与函数 contour 相同。

【例 3-27】绘制填充的二维等高线图形。

```
[X,Y,Z]=peaks;
contourf(Z,20)
```

其执行结果如图 3-28 所示。

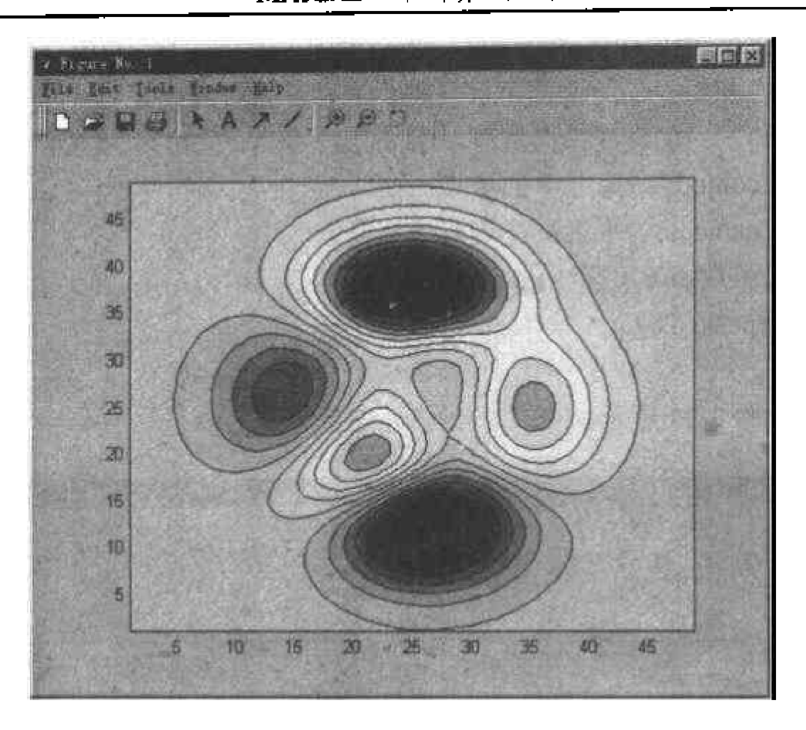


图 3-28　填充的二维等高线图

3.3.6　三维视图可视效果的控制

由于三维视图表现的是一个空间图形，因此，从不同的位置和角度观察图形有不同的效果。另外，在复杂的三维图形中，经常会出现图形的某一部分被遮住的情况，这会对一些问题的分析造成严重的影响。针对这一问题，在 MATLAB 中创建了对视图可视效果控制的命令函数。

1. 三维图形观察点和视觉的控制 view

在 MATLAB 中，控制图形观察点和视觉的函数是 view，其使用格式如下。

- view(AZ,EL)和 view([AZ,EL])　通过方位角和俯视角设置观察图形的视点。其中：AZ 为方位角（Azimuth），指在 x-y 平面内从 y 轴负方向绕 z 轴旋转的角度，以逆时针为正；EL 为俯视角（Elevition），指在 x-y 平面沿 z 轴方向仰起的角度。
- view([*X Y Z*])　通过直角坐标系设置视点。
- [AZ,EL] = view　返回当前的方位角 AZ 和俯视角 EL。
- view(*T*)　用一个 4×4 的转矩阵 *T* 来设置视角。
- T=view　返回当前的 4×4 的转矩阵。

注意：

view(2)格式　设置缺省的二维视角。二维图形缺省值为 AZ = 0、EL = 90。

view(3)格式　设置缺省的三维视角。三维图形缺省值为 AZ = -37.5、EL = 30。

从 MATLAB 5.3 版本起，其图形窗口可交互式调节视点。为获得最佳的视觉效果，用户可先通过鼠标操作调节视点，然后再用命令 view 把相应的视点加以固定。

若在 MATLAB 5.3 以前的版本中调节视点，必须先运作 rotate3d 后，才可用鼠标进行交互式调节。

【例 3-28】练习使用函数 view 的用法。

```
[X,Y]=meshgrid(-8:0.5:8);
R=sqrt(X.^2+Y.^2)+eps;
Z=sin(R)./R;
subplot(2,2,1)    %缺省视角
surf(X,Y,Z)
xlabel('X 轴','FontWeight','bold')
ylabel('y 轴','FontWeight','bold')
zlabel('z 轴','FontWeight','hold')
title('\fontname {隶书} 缺省视角')
subplot(2,2,2)
surf(X,Y,Z)
xlabel('X 轴','FontWeight','bold')
ylabel('y 轴','FontWeight','bold')
zlabel('z 轴','FontWeight','bold')
title('\fontname {隶书} 方位角为 90°  仰角为 0° ')
view(90,0)
subplot(2,2,3)
snrf(X,Y,Z)
xlahel('X 轴','FontWeight','bold')
ylabel('y 轴','FontWeight','bold')
zlabel('z 轴','FontWeight','bold')
title('\fontname {隶书} 方位角为-37.5° 仰角为 80° ')
view(-37.5,80)
subplot(2,2,4)
surf(X,Y,Z)
xlabel('X 轴','FontWeight','bold')
ylabel('y 轴','FontWeight','bold')
zlabel('z 轴','FontWeight','bold')
title('\fontname {隶书} 方位角为 0° 仰角为 90° ')
view(0,90)
```

其执行结果如图 3-29 所示。

图 3-29 中左上角子窗口就是平常缺省设置；右上角子窗口是把方位角在 x-y 平面内从 y 轴负方向逆时针旋转 90°，转到 x 轴的正方向，俯视角为 0°，即视线从 x 轴的正方向水平看过去的效果；左下角子窗口把方位角恢复为缺省值，俯视角为 80°，即从斜上方往下看时的效果图；右下角子窗口把方位角设置为 0°，而俯视角为 90°，即从正上方往下看时的效果图。

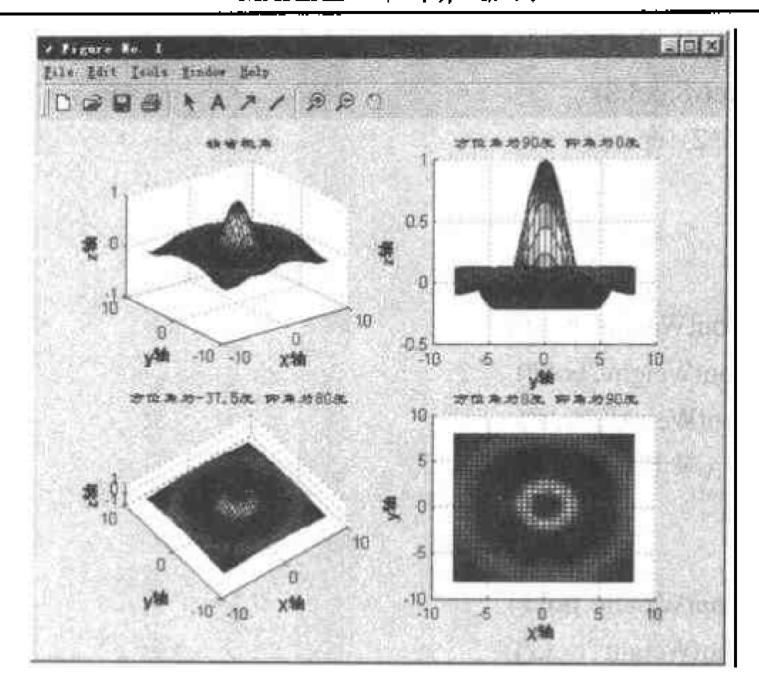


图 3-29 view 函数使用效果比较图

2. 三维图形的照相 campos

在 MATLAB 中有一种类似照相机可变焦透镜功能的函数，以实现对观察点的控制。此种功能的函数比较多，我们仅以函数 campos 来说明。

其调用格式为：

- CP = campos 得到当前句柄照相机位置；
- campos([*X Y Z*]) 设置照相机位置；
- campos(mode) 得到照相机位置模式。

【例 3-29】照相技术的应用。

```
[X,Y]=meshgrid(-8:0.5:8);
R=sqrt(X.^2+Y.^2)+eps;
Z=sin(R)./R;
surf(X,Y,Z)
xlabel('X 轴','FontWeight','bold')
ylabel('y 轴','FontWeight','bold')
zlabel('z 轴','FontWeight','bold')
title('\fontname {隶书} 照相技术应用')
campos
campos([36.5,49.0,-10])
```

结果为：

```
ans =
    -91.3142   -119.0030      6.7452
```

其运行结果如图 3-30 所示。

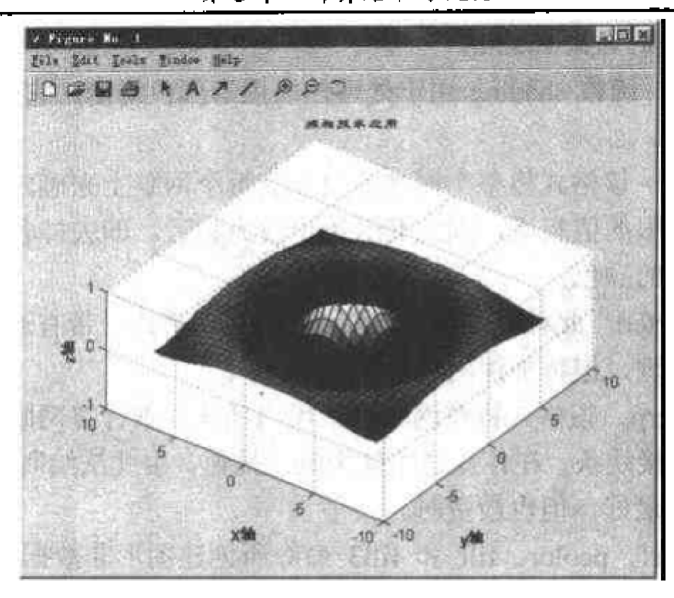


图 3-30　照相技术的应用

3. 三维图形的透视命令 hidden

在 MATLAB 中，用 mesh 命令绘制网格图时，在默认的情况下系统会消隐掉重叠在后面的网格，而利用透视命令 hidden 则可以看到被掩盖的部分。

hidden 命令的使用很简单，它只是开关掩盖命令。其调用命令格式为：

- hidden on　默认模式，消隐掉后面的网格线；
- hidden off　关掉消隐命令，从而能看到被遮挡的部分。

【例 3-30】hidden off 模式的图形。

```
mesh(peaks(20))
hidden off
```

其执行结果如图 3-31 所示。

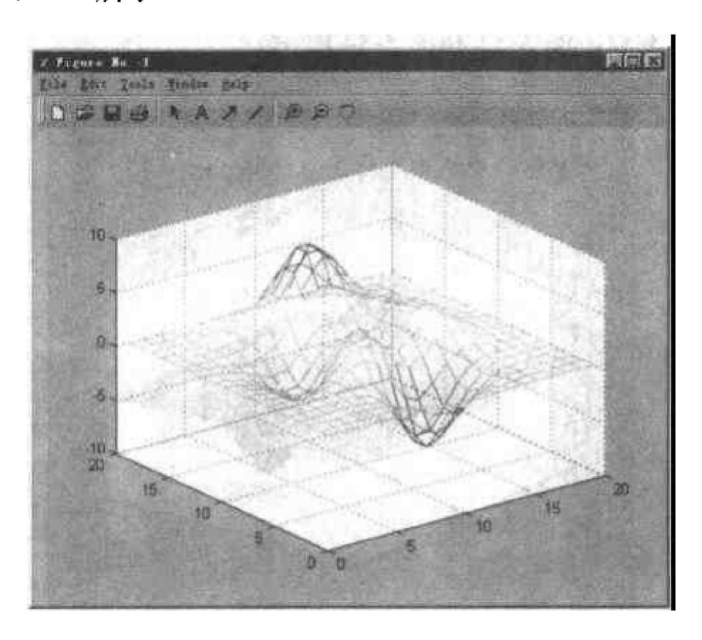


图 3-31　网格消隐图

4. 曲面图形颜色的遮掩 shading

在 MATLAB 中，函数 shading 用于处理曲面图形颜色均衡。它有三种浓淡处理方式，其调用格式分别如下。

- shading flat　该格式将整个网格的值在其每个网眼上来确定一个标志颜色的值，由于相邻网眼的值相近，因此其颜色也较为相近。即去掉各片连接处的线条，平滑当前图形的颜色；
- shading faceted　此种格式是 MATLAB 的默认格式，带有连接线条，它对网眼的颜色不做处理，但将加深网线的黑色；
- shading interp　该格式将在网眼内采用内插法详细计算网眼内不同位置的颜色差异，去掉连接线条，在各片之间使用颜色插值，由此法绘制的图形，颜色最连贯，着色光顺性最好，但也最费时。

注意：mesh、surf、pcolor、fill 和 fill3 命令所创建图形非数据点处的着色由 shading 命令决定。

下面举一个例子说明它们的用法。

【例 3-31】函数 shading 三种浓淡处理方式比较。

```
Z=peaks(60);
surf(Z)
shading flat
figure
surf(Z)
shading faceted
figure
surf(Z)
shading interp
```

其执行结果如图 3-32、图 3-33 和图 3-34 所示。

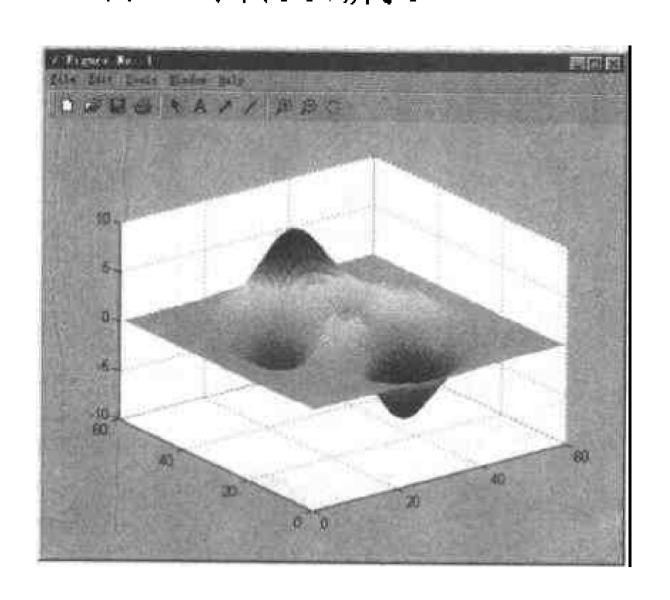


图 3-32　shading flat 效果图

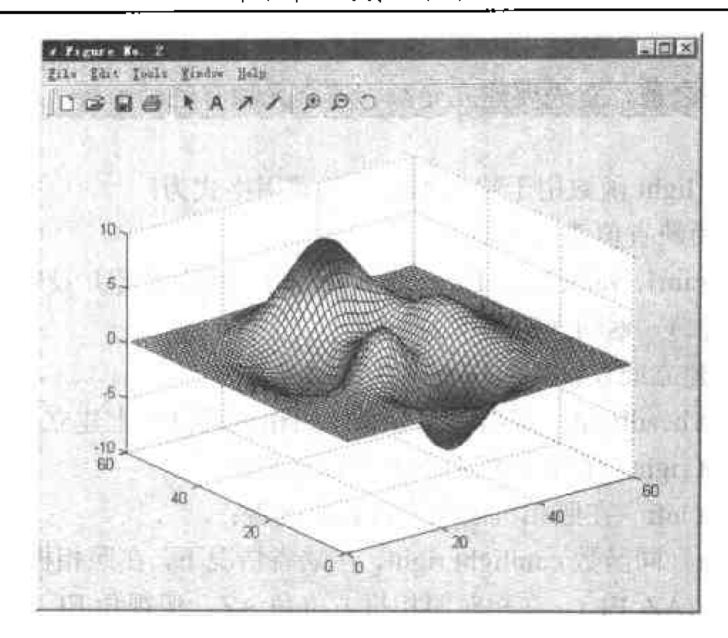


图 3-33　shading faceted 效果图

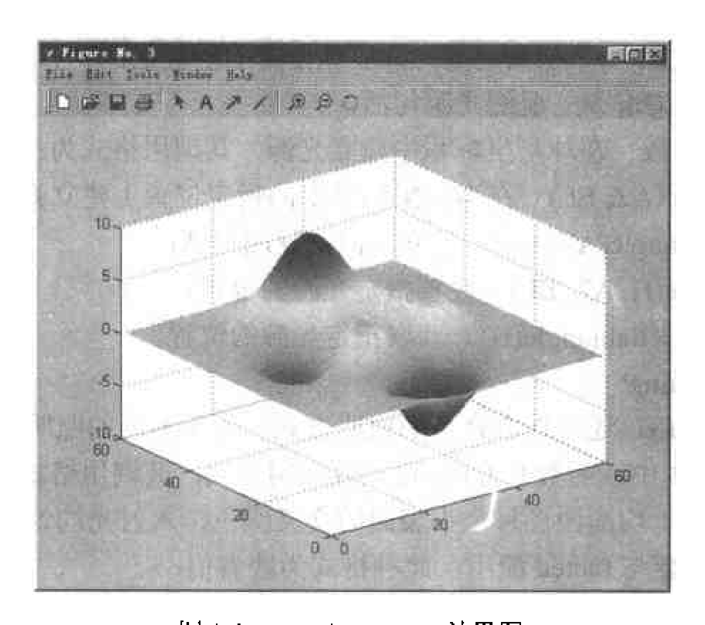


图 3-34　shading interp 效果图

3.3.7　三维图形的照明和材质处理

为了把图形表现得更加真实，MATLAB 提供了光源设置、照明方式和反射光处理的“高层”命令函数，其函数名称和含义如表 3-2 所示。

表 3-2　　照明及光源设置函数

函数	功能
camlight	根据照相机的位置建立或移动光源
light	建立光源对象
lightangle	在球形坐标系中建立或放置光源
lighting	选择照明方式
material	设置目标的反射

1. 光源的建立

在设置光照效果之前，首先要建立光源。建立光源有三种函数，即 light、camlight 和 lightangle 函数。

- light 函数 light 函数用于建立光源，其调用格式为：
  - ◆ light 用缺省值建立光源；
  - ◆ light(Param1, Value1, ..., ParamN, ValueN) 建立光源并设置参数；
  - ◆ L=light(...) 返回光源对象的句柄。
- camlight 建立或设置光源位置，其调用方式为：
  - ◆ camlight headlight 在当前坐标系中照相机的位置上建立光源；
  - ◆ camlight right 在照相机的右上方建立光源；
  - ◆ camlight left 在照相机的左上方建立光源；
  - ◆ camlight 同函数 camlight right，在缺省情况下，在照相机的右上方建立光源；
  - ◆ camlight(AZ, EL) 在相对照相机方位角 AZ，俯视角 EL 的位置上建立光源；
  - ◆ camlight(..., style) 设置光源的类型，类型可以是“local”（缺省值）或“infinite”；
  - ◆ camlight(H, ...) 把指定的光源放在指定的位置上；
  - ◆ H = camlight(...) 返回光源句柄。
- lightangle 函数 在球形坐标系中建立光源．其调用格式为：
  - ◆ lightangle(AZ, EL) 在当前坐标系中的指定位置上建立光源；
  - ◆ H = lightangle(AZ, EL) 建立光源并返回句柄；
  - ◆ lightangle(H, AZ, EL) 设置指定光源的位置；
  - ◆ [AZ,EL] = lightangle(H) 获取指定光源的位置。

2. 照明方式 lighting

为图形设置光源后，还要选取合适的照明方式，因为不同的照明方式，图形的效果也不一样。在 MATLAB 中，函数 lighting 可设置照明方式，其调用格式为：

- lighting flat 均衡图形每个小表面的交叉颜色，入射光均匀洒落在图形对象的每个面上，主要与 facted 配用，此种格式为缺省值；
- lighting ground 对小表面的交叉颜色作颜色插值，先对顶点颜色插值，再对顶点勾画的面色进行插值，用于曲面表现；
- lighting phong 对小表面的交叉颜色作颜色插值，先对顶点处法线插值，再计算每个像素的反光。此种格式的表现效果最好，但费时较多；
- lighting none 关闭所有光源。

注意：*lighting none 命令只有在 light 等设置光源命令执行后才起作用；命令 material 只能对由函数 surf、mesh、pcolor、fill 和 fill3 生成的曲面起作用。*

3. 控制光效果的材质命令 material

设置光源和照明方式后，为了取得更为逼真的图形效果，在 MATLAB 中又提供了控制光效果的材质命令函数 material，其调用格式为以下几种。

- material shiny 使对象比较明亮。镜反射份额较大，反射光颜色仅取决于光源的颜色。

- material dull 使对象比较暗淡。漫反射份额较大，没有镜面亮点，反射光颜色仅取决于光源颜色。
- material metal 使对象带金属光泽。镜反射份额较大，背景光和漫反射份额很小。反射光颜色取决于光源和图形表面的颜色。该镜式为缺省格式。
- material default 返回缺省的模式。
- material([ka kd ks n sc]) 对五大反射要素进行设置。其中：
  - ◆ ka 设置无方向性的、均匀的背景光（Ambient Light）强度；
  - ◆ kd 设置无方向性的软反射的漫反射（Diffuse Reflection）强度；
  - ◆ ks 设置有硬反射光（Specular Reflection）的强度；
  - ◆ n 设置控制镜面亮点大小的镜面指数（Specular Exponent）；
  - ◆ sc 控制镜面颜色的反射系数（Specular ColorReflectance）。

下面我们举一个综合的例子说明它们的用法。

【例 3-32】绘制一个球体，使用照明和材质处理指令，并比较处理后的效果。

```
subplot(2,2,1)
sphere
axis equal
shading faceted
camlight left
lighting none
title('第一子图')
subplot(2,2,2)
sphere
axis eqnal
shading flat
camlight right
camlight left
lighting flat
title('第二子图')
subplot(2,2,3)
sphere
axis equal
shading interp
camlight right
camlight left
lighting gouraud
material([0.5,0.3,0.5,10,0.5])
title('第三子图')
subplot(2,2,4)
sphere
```

```
axis equal
shading interp
light('position',[-1,-1,-2],'color','y')
light('position',[-1,0.5,1],'style','local','color','r')
lighting phong
material([0.4,0.5,0.3,10,0.3])
title('第四子图')
```

其执行结果如图 3-35 所示。

从图 3-35 中可以看出：每个子图可以定义自己的浓淡处理模式、照明模式和材质，但它们都只能定义一次；每个子图可以设置多个光源；设置方式可以是多种。

在 material 命令中，ka、kd、ks 对亮度影响很大。数值愈大，光愈强；参数 *n* 愈大，镜面亮点的范围愈小。

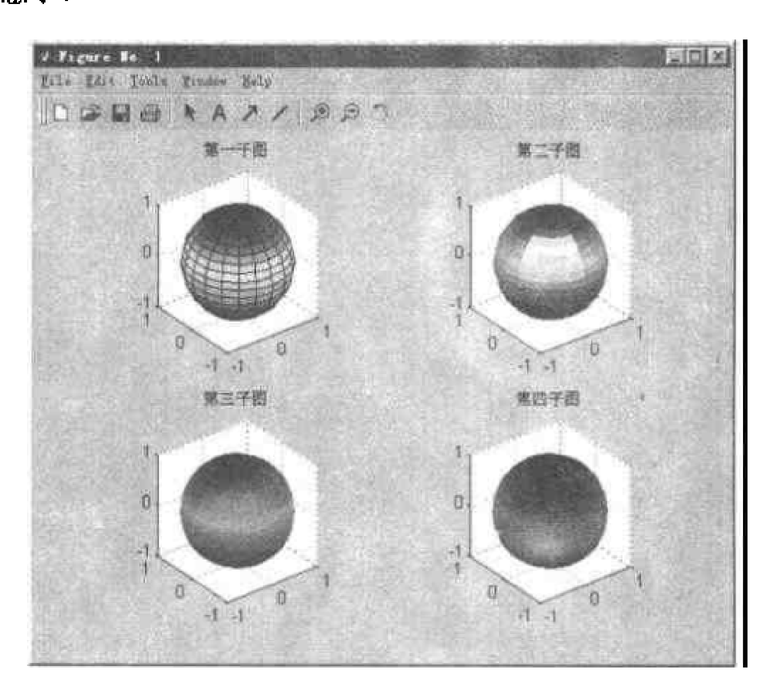


图 3-35 使用照明和材质命令所表现的图形

3.3.8 柱面和球面的三维表达

在 MATLAB 中，有两个用来绘制柱面和球面的命令，即 cylinder 和 sphere 函数命令。它们可以非常简洁地绘制漂亮的彩图。

1. 柱面的表达

与 AotoCAD 一样，MATLAB 绘制一个柱面首先要给出它的母线和轴线。而在 cylinder 命令中，柱面的轴线已经定义为 *z* 轴，因此只要给出母线的描述就可以完成一个柱面，再加上参数就可完成一个完整的柱面。其调用格式为：

- [*X*,*Y*,*Z*] = cylinder(*R*,*N*)，其中：
  - ◆ *R* 是一描述柱面母线的向量；
  - ◆ *N* 是旋转柱面上的分割线条数；

◆ [X，Y，Z]是 cylinder 命令执行后，返回的 x、y、z 坐标向量。可以用 surf(X, Y, Z)函数命令显示柱面。

- [X,Y,Z] = cylinder(R)　缺省值 N=20。
- [X,Y,Z] = cylinder　缺省值 N=20，R=[1，1]。

【例 3-33】绘制一个柱面。

```
t=pi:0.01:3*pi;
r=sin(t)+t;
cylinder(r,30)
shading interp
```

其执行结果如图 3-36 所示。

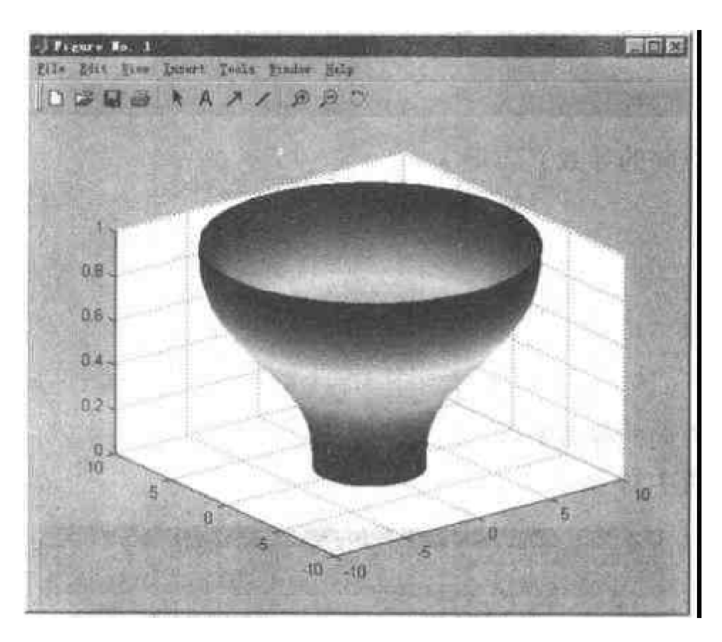


图 3-36　柱面图

2. 球面的表达

由于绘制球体只是一个单位球面，因此比绘制柱面要简单，其调用格式为：

- [X,Y,Z] = sphere(N)　此种格式将产生一个三维的（n+1）×（n+1）的矩阵，然后用函数 surf 命令绘制一个单位的球面；
- [X,Y,Z] = sphere　此种格式中，其缺省值 N = 20。

其中 N 表示设置分割线的条数。

由于此命令很少单独使用，常和其他的命令一起使用，并且使用简单，此处不再举例。

3.4　四维表现图

人对自然界的理解是多维的。人不仅接受一维、二维、三维的几何信息，而且对几何物体的运动以及颜色等反应灵敏。因此，对于一般定义在 x-y-z 坐标系上的四维可视化，

MATLAB 的色彩控制和动画等命令可为四维甚至更高维的表现提供实现手段。

3.4.1 用色彩进行四维表现

当三维网格线图和曲面图的第四个输入参数取一些特殊的矩阵时，色彩就能表现出函数的某些特征。

【例 3-34】用色彩阵表现函数的四维。

```
[X,Y,Z]=peaks(30);
[dzx,dzy]=gradient(Z);
subplot(1,2,1)
surf(X,Y,Z,dzx)
shading faceted
brighten(0.1)
colorbar('horiz')
title('函数 x 轴方向的导数')
subplot(1,2,2)
surf(X,Y,Z,dzy)
shading faceted
brighten(0.1)
colorbar('horiz')
title('函数 y 轴方向的导数')
```

其执行结果如图 3-37 所示。

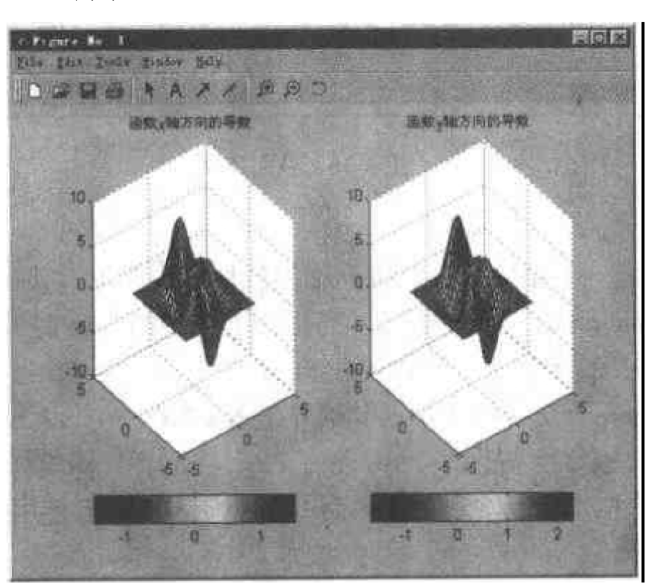


图 3-37 色彩阵进行四维表现

3.4.2 用切片图和等位线图进行四维表现

在 MATLAB 中，提供了函数 slice 和函数 contourslice 来绘制三维物体切片图，用于三元函数 $R=f(x,y,z)$的四维可视化。

函数 slice 的调用命令格式为：

- [*X*，*Y*，*Z*]=meshgrid(*x*，*y*，*z*) 用于三维网格坐标的生成；
- slice（*X,Y,Z,V,Sx,Sy,Sz*） 用于绘制三元函数切片图采用 slice(*V,Sx,Sy,Sz*) 或 slice(*V,XI,YI,ZI*)格式时，MATLAB 假设 *X*=1:*N*, *Y*=1:*M*, *Z*=1:*P*。其中：
  - ◆ *x*、*y*、*z* 是各自变量的分度向量，决定“网线”的位置，分别是（$1\times n$）、（$1\times m$）和（$1\times p$）的向量；
  - ◆ *X*、*Y*、*Z* 是三维网格坐标，它们是（$m\times n\times p$）维数组（注意维数的次序）；
  - ◆ *V* 是网线节点上的三元函数值数组，维数为（$m\times n\times p$）；
  - ◆ *Sx*、*Sy*、*Sz* 决定切片位置的数值向量，即分别决定垂直于 *x*、*y*、*z* 轴切面的位置向量。它们的维数可以不同，假如取 0 维空阵时，表示没有切片存在。

切片上的函数值大小可以用色轴上的对应颜色表示。*V* 数组中的最大有限值和最小有限值定义了色轴的范围。由于切片的位置可以任意设置，因此 slice 通过三维坐标点上的色彩变化把图形的表现能力扩展到四维。

【例 3-35】绘制函数 $V=x\times e^{-(x^2+y^2+z^2)}$ 的四维表现图，其中 *x*，*y*，*z* 在区间（-2，2）内，切片上的函数值用色轴上对应的颜色表示。

```
[x,y,z] = meshgrid(-2:.2:2, -2:.25:2, -2:.16:2);
v = x .* exp(-x.^2 - y.^2 - z.^2);
slice(x,y,z,v,[-1.2 .8 2],2,[-2 -.2])
xlabel('x 轴')
ylabel('y 轴')
zlabel('z 轴')
colorbar('horiz')
view(-30,45)'
```

其执行结果如图 3-38 所示。

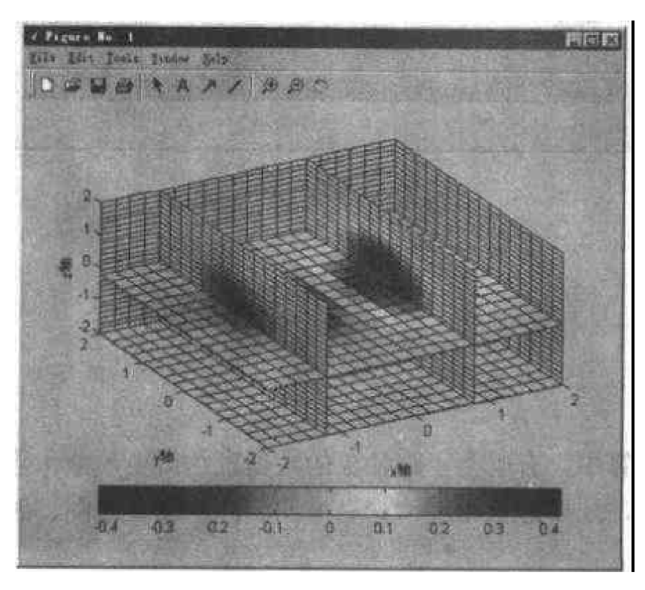


图 3-38 切片图

切片等位线图是在切片内绘制的等位线图，其调用格式为包括以下几点：

- comtourslice(*X*,*Y*,*Z*,*V*,*Sx*,*Sy*,*Sz*) 这种格式中的参数与 slice 相同；
- contourslice(*V*,*Sx*,*Sy*,*Sz*)或 contourslice(*V*,*XI*,*YI*,*ZI*) 调用此种格式时，MATLAB 假设[*X Y Z*] = meshgrid(1:*N*, 1:*M*, 1:*P*)，这里 [*M*,*N*,*P*]=SIZE(*V*)；
- contourslice(..., *N*) 调用此种格式时，则在每个切片平面内绘制 *N* 条等位线；
- contourslice (..., CVALS) CVALS 是决定等位线的函数值采样向量。

【例 3-36】绘制切片等位线图。

```
[x y z v] = flow;
h=contourslice(x,y,z,v,[1:9],[],[0], linspace(-8,2,10));
axis([0 10 -3 3 -3 3]);
view([-12,30])
colormap jet
colorbar
box on
```

其执行结果如图 3-39 所示。

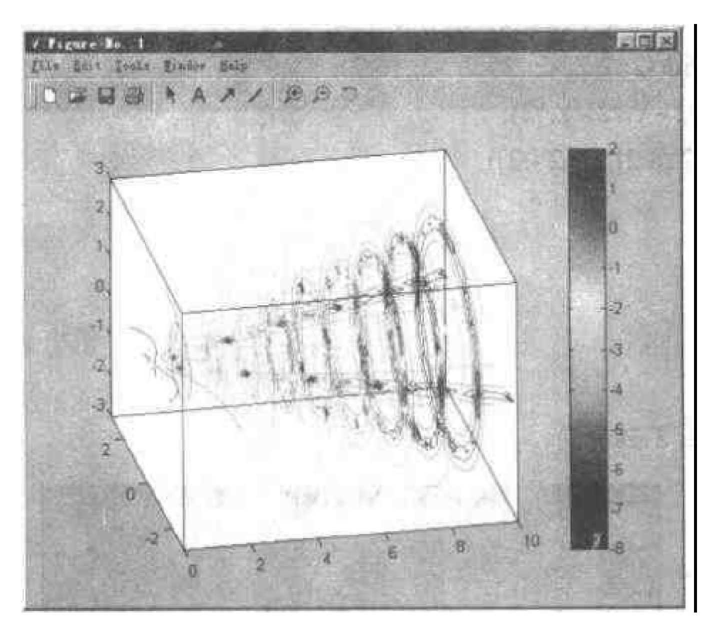


图 3-39 切片等位线图

3.5 特殊图形

在工程教学和计算中，有时为了将抽象的数据表达得更形象，除了绘制出它的二维、三维以及四维图形外，还要经常用到诸如直方图、面积图、饼图、射线图、阶梯图以及火柴图等特殊图形。

MATLAB 提供了很多绘制特殊图形的命令，如表 3-3 所示。

表 3-3 绘制特殊图形指令

函数名称	功 能	函数名称	功 能
area	面积图	compss	指针图
bar	竖直的直方图	hist	向量的统计直方图
barh	水平直方图	pareto	带有标准的直方面
bar3	三维竖直直方图	pie	二绘饼图
bar3h	三维水平直方图	pie3	三维饼图
gplot	拓扑图	plotmatrix	矩阵折（曲）线图
comet	彗星轨迹状的图形	ribbon	带状图
errorbar	误差棒图	scatter	散点图
ezplot	符号函数二维曲线图	stem	火柴杆图
polar	极坐标曲线图	stem3	三维火柴杆图
feather	羽毛图	stairs	阶梯图

3.5.1 面积图命令 area

在实际应用中，面积图适用于表现各个不同部分对整体所作的贡献，因此适用范围很广。其调用格式为：

- area(*X*,*Y*) 该格式的使用与 plot 的命令的使用方法一样，只不过在绘制过程中，area 命令将连线图到 *x* 轴的那部分填上了颜色；
- area(*Y*) 该格式的使用与 area(*X*,*Y*)一样，只是此时缺省值 X=1:SIZE(*Y*,1)；
- area(*X*,*Y*,LEVEL)或 area(*Y*,LEVEL) 此种格式与 area(*X*,*Y*)、area(*Y*)不同的是填色部分改为由连线图到 *y*=level 的水平线之间的部分。

【例 3-37】绘制一个面积图。

程序命令如下：

```
X=-2:2;
Y=[3,5,2,4,1;5,4,2,3,5;3,4,5,2,1];
area(X',Y')
legend('因素 1','因素 2','因素 3')
grid on
colormap(spring)
```

其运行结果如图 3-40 所示。

3.5.2 直方面命令 bar

直方图常用于统计数据的作图，由于函数 bar、bar3、barh 和 bar3h 的调用格式类似，以函数 bar 为例，其调用格式为：

- bar(*X*,*Y*) *X* 是横坐标向量，*Y* 可以是向量或矩阵。*Y* 是向量时，每一个元素对应一个竖条；*Y* 是 *m* 行 *n* 列矩阵时，将画出 *m* 组竖条，每组包括 *n* 竖条；
- bar(*Y*) 横坐标使用缺省值 *X*=1:*M*；
- bar(*X*,*Y*,WIDTH) 或 bar(*Y*,WIDTH) 用 WIDTH 指定竖条的宽度，如果 WIDTH ＞1，条与条之间将重合，缺省宽度为 0.8；
- bar(...,'grouped') 产生缺省的组合直方图；

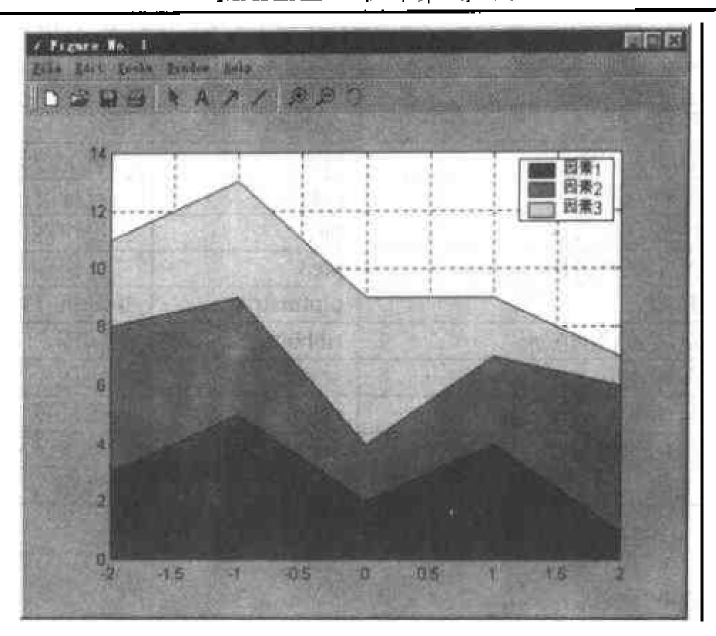


图 3-40　面积图

- bar(...,'stacked')　产生的直方图；
- bar(...,linespec)　指定条的颜色；
- H = bar(...)　返回补片对象的句柄。

下面我们通过例子来说明其用法。

【例 3-38】用绘制直方图的几种命令绘制直方图。

```
X=-2:2;
Y=[3,5,2,4,1;5,4,2,3,5;3,4,5,2,1];
subplot(2,2,1)
bar(X,Y','r')
xlabel('x')
ylabel('y')
colormap(cool)
subplot(2,2,2)
barh(X,Y','grouped')
xlabel('y')
ylabel('x')
colormap(cool)
subplot(2,2,3)
bar(X,Y','stacked')
xlabel('x')
ylabel('\Sigma y')
colormap(spring)
subplot(2,2,4)
barh(X,Y','stacked')
```

```
xlabel('y')
ylabel('\Sigma x')
colormap(summer)
```

其执行结果如图 3-41 所示。

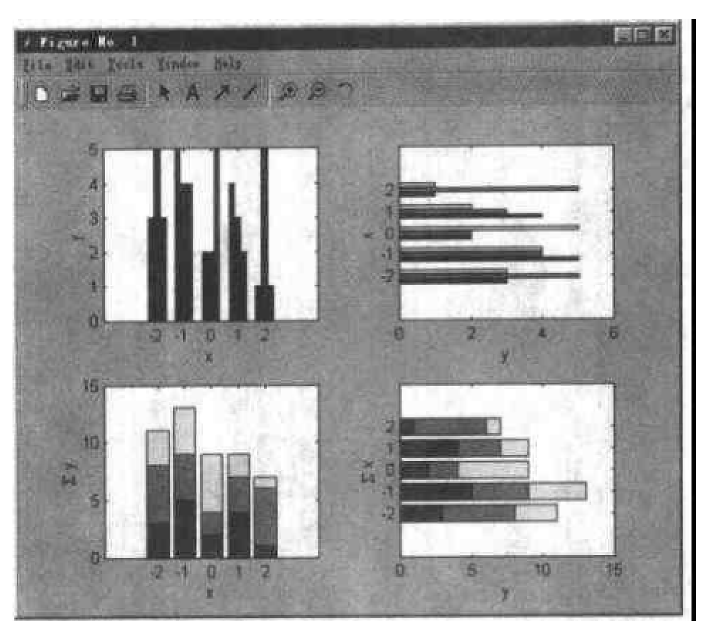


图 3-41 二维直方图

【例 3-39】绘制三维直方图。

```
X=-2:2;
Y=[3,5,2,4,1;5,4,2,3,5;3,4,5,2,1];
subplot(2,2,1)
bar3(X,Y','r')
zlabel('y')
ylabel('x')
colormap(cool)
subplot(2,2,2)
bar3h(X,Y','grouped')
ylabel('x')
zlabel('y')
colormap(cool)
subplot(2,2,3)
bar3(X,Y','stacked')
ylabel('x')
zlabel('\Sigma y')
colormap(spring)
subplot(2,2,4)
```

```
bar3h(X,Y','stacked')
zlabel('x')
ylabel('\Sigma y')
colormap(summer)
```

其执行结果如图 3-42 所示。

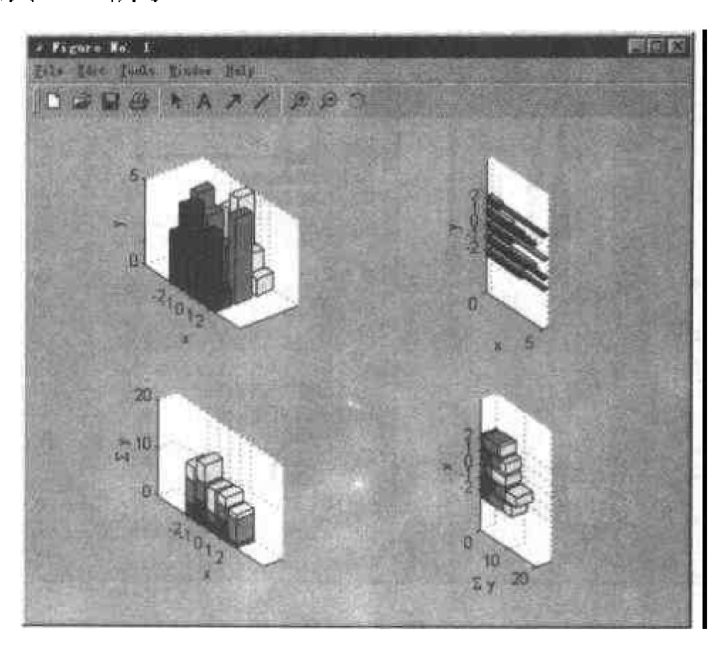


图 3-42　三维直方图

注意：画好的直方图同样可以用 set 和 get 函数设置其状态，而且还可以用 shading 命令设置颜色的平滑方式。

3.5.3　饼图命令 pie

饼图又叫扇形图，主要用于显示向量中元素所占向量元素总和的百分比。在 MATLAB 中实现此功能的是函数 pie 和 pie3，它们分别绘制二维和三维饼图。因为二者的用法相似，仅以函数 pie 为例，说明它们的调用格式：

- pie(*X*)　绘制向量 *X* 的饼图。该命令将把 *X* 的每一个元素在所有元素总和中占的比例表达出来；
- pie(*X*,EXPLODE)　向量 EXPLODE 用于指定饼图中抽出一部分的块。它和向量 *X* 必须长度相等，向量 EXPLODE 中的非零值对应的块将被抽出；
- pie(...,LABELS)　LABELS 是用于标注饼图的字符串数组，其长度必须和向量 *X* 相等；
- *H* = pie(...)　返回包括补片和文本对象句柄的向量。

【例 3-40】用函数 pie 和 pie3 绘制饼图。

```
x=[200,360,120,400,320];
subplot(2,2,1),pie(x,[0 0 0 1 0])
subplot(2,2,2),pie3(x,[0 0 0 1 0])
subplot(2,2,3),pie(x(2:5))
```

```
subplot(2,2,4), x=[0.1,0.12,0.21,0.34,0.11];pie3(x ,{'A','B','C','D','E'})
```

其执行结果如图 3-43 所示。

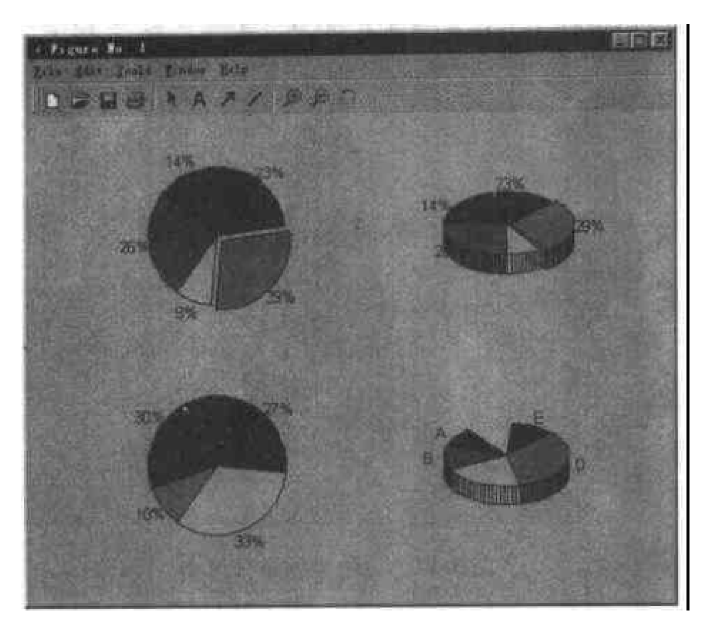


图 3-43　饼图

从图 3-43 第四幅图中我们得知，当 x 的所有元素之和不足为 1 时，饼图命令 pie 将把一个元素值作为其在饼图中所占比例绘制，得到的结果是一个不完整的饼图。

3.5.4　柱形图命令 hist

柱形图和直方图的形状类似，但作用不同，它主要用于显示数据的分布规律。在 MATLAB 中用于建立柱形图的函数有 hist 和 rose，hist 是在直角坐标系中建立柱形图，而 rose 是在极坐标系中建立柱形图，现仅以 hist 为例来说明它们的调用格式：

- hist(*Y*)　把 *Y* 按其中数据的大小分为 10 个长度相等的段，统计每段中的元素个数，并返回给 *N*，如果 *Y* 是矩阵，那么按列分段；
- *N*=hist(*Y*,*M*)　*M* 是标量，设置分段数，由自己设定将 *Y* 的取值范围[*a*,*b*]划分为 *M* 等份，同样以 *N* 作为返回值；
- *N*=hist(*Y*,*X*)　*X* 是向量，将以向量 *X* 的每个元素值为中心，统计出在每个中心附近 *Y* 的元素个数，并将其返回给 *N*；
- [*N*,*X*] = hist(...)　同时返回每个数据段中间值的位置；
- hist(...)　不带输出参数，直接绘制柱形图。

【例 3-41】绘制柱状图。

```
x=-4:0.1:4;
y=randn(10000,1);
hist(y,x)          %绘制一个直角坐标系下的柱状图
figure(2)
theta=y*2*pi;      %绘制一个极坐标系下的柱状图
rose(theta)
```

其执行结果如图 3-44、图 3-45 所示。

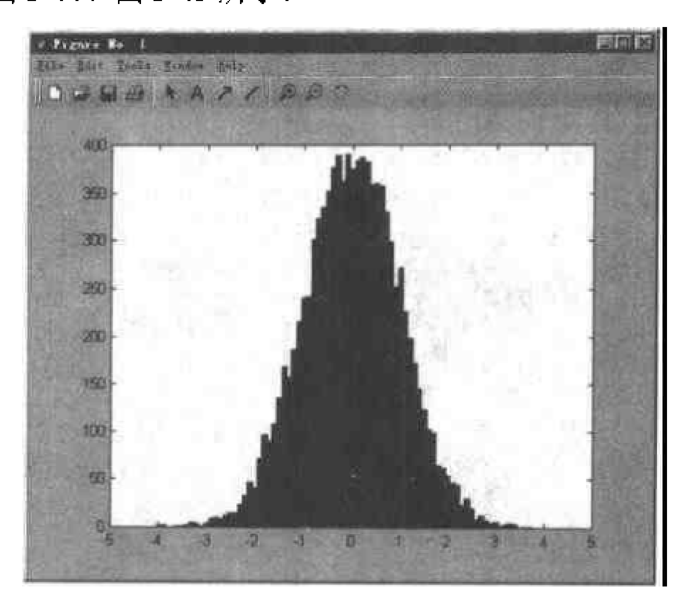


图 3-44 直角坐标系柱形图

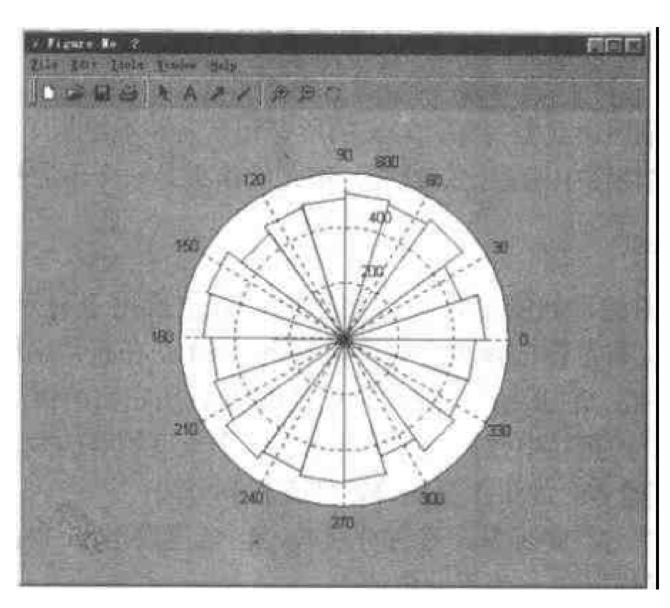


图 3-45 极坐标系柱形图

3.5.5 火柴杆图命令 stem

在工程教学和实践中，会经常遇到离散数据的绘图问题。MATLAB 提供了一些操作简单而又实用的指令，如函数 stem 等。

函数 stem 和 stem3 分别用来绘制二维和三维的离散图形，其用法与 plot 和 plot3 基本一致。主要区别是 stem 和 stem3 中可以用选项 “filled” 来填充图中离散点的标记，即将图中的“火柴头”填上颜色。

下面通过例 3-42 来说明如何绘制二维和三维火柴杆图。

【例 3-42】绘制二维和三维火柴杆图。

x=0:0.1:4;

```
y=sin(x).*exp(cos(x));
stem(x,y,'rp')                    %绘制二维图形
figure(2)
y=0:0.1:4;
z=x.^2+y.^2;
stem3(cos(x),exp(y),z,'bd')       %绘制三维图形
```

编制 M 文件 C3L42，其运行结果如图 3-46、图 3-47 所示。

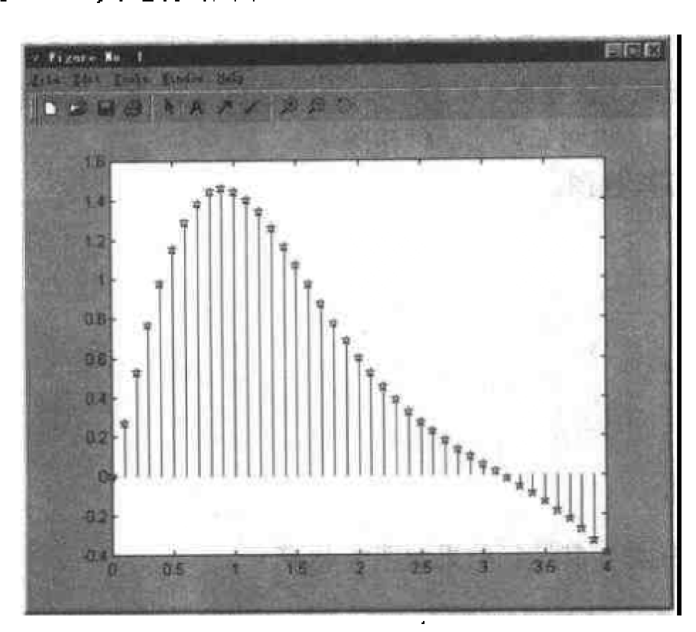


图 3-46 二维火柴杆图

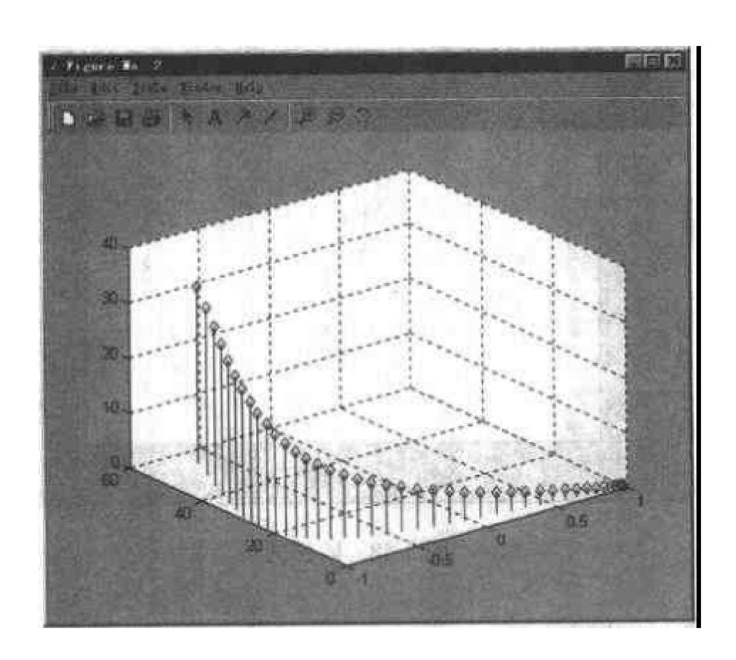


图 3-47 三维火柴杆图

3.5.6 阶梯图命令 stairs

阶梯图与火柴杆图类似，均是表现间断数据的工具。函数 stairs 用于绘制类似楼梯形状的步进图。其调用格式为：

stairs(*Y*)

stairs(*X*,*Y*)

stairs(...,STYLE)

[*XX*,*YY*] = stairs(*X*,*Y*)

前三种调用格式与 stem 命令格式的使用一样。第四种格式并不画出阶梯图，而是将阶梯图上各点坐标值赋予变量 *XX* 和 *YY*，然后可用 plot(*XX*,*YY*)命令绘制阶梯图。

下面将通过例 3-43 说明如何绘制阶梯图。

【例 3-43】绘制阶梯图。

程序的内容如下：

```
x=0:0.2:2*pi;
y=sin(x);
stairs(x,y,'g')
hold on
plot(x,y,'r:')
hold off
```

编制 M 文件 C3L43，其运行结果如图 3-48 所示。

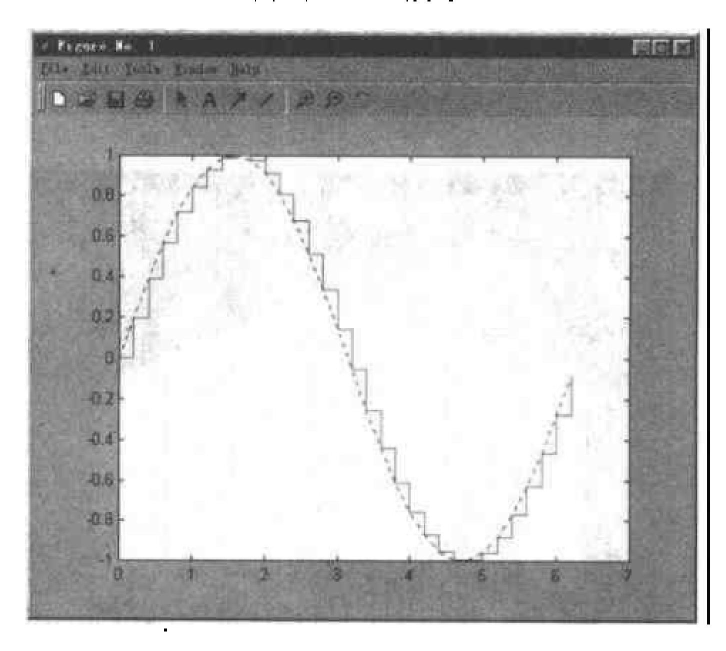


图 3-48 阶梯图

3.5.7 误差棒图命令 errorbar

误差棒图适用于表现数据的可信度或关于一个已知标准的偏离程度。其调用格式为：

errorbar(*X*,*Y*,*L*,*U*)

errorbar(*X*,*Y*,*E*) 或 errorbar(*Y*,*E*)

errorbar(...,'LineSpec')

- *X*，*Y*，*E*，*L*，*U* 是长度相等的向量。如是矩阵，则按每列绘制误差棒。
- *L* 和 *U* 分别是指定在向量 *Y* 中每点的误差范围的上限和下限，而每一误差棒的长度为 $L(i) + U(i)$。

第二种格式的作用是 plot(*Y*)，在此基础上，图中每个元素所对应点上画出一根误差棒，误差棒的长度为 $2E(i)$，即误差棒的中点为（*X*,*Y*）。

【例 3-44】绘制误差棒图。

```
x = 1:10;
y = sin(x);
e = std(y)*ones(size(x));
errorbar(x,y,e)
```

其执行结果如图 3-49 所示。

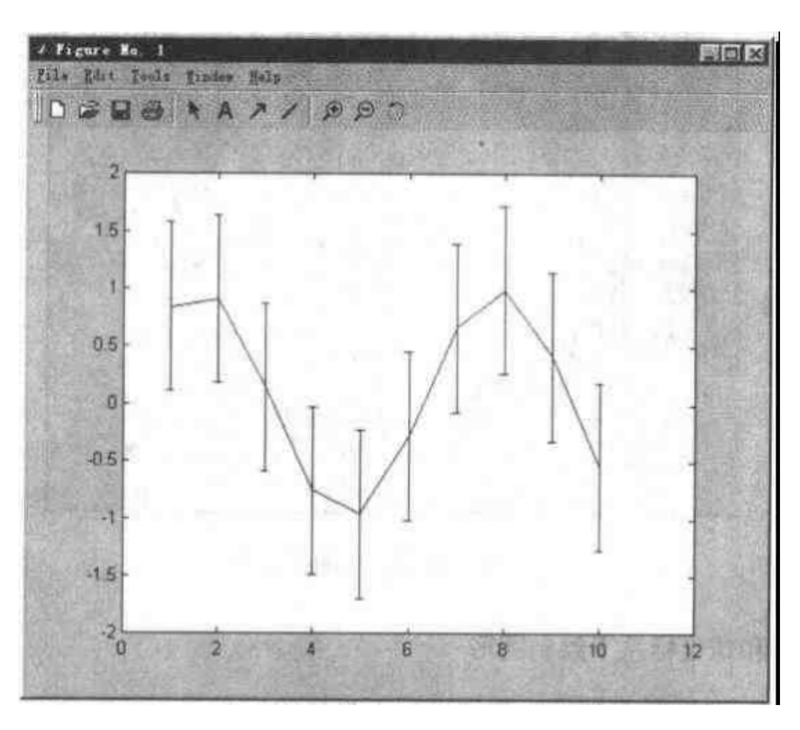


图 3-49　误差棒图

3.6　坐标系下绘制二维和三维图形

在 MATLAB 中的绘图命令不仅能在直角坐标系下绘制图形，而且也能在极坐标、柱坐标和球坐标系下绘制图形。

3.6.1　极坐标系下绘制图形

在 MATLAB 中提供了直接在极坐标系下绘图的命令 ploar，它可以直接把极坐标系下

表达的坐标值矩阵绘制成连线图，而且同样可以用“linestyle”字符来控制连线图的线型。其调用格式为：

ploar(THETA, RHO,*S*)

其中 *S* 是字符串，用来控制图形的线型。

【例 3-45】绘制半径为 2 的渐开线。

```
rhe=2;
theta=0:pi/20:4*pi;
rho=rhe+theta*rhe;
polar(theta,rho,'r')
```

其执行结果如图 3-50 所示。

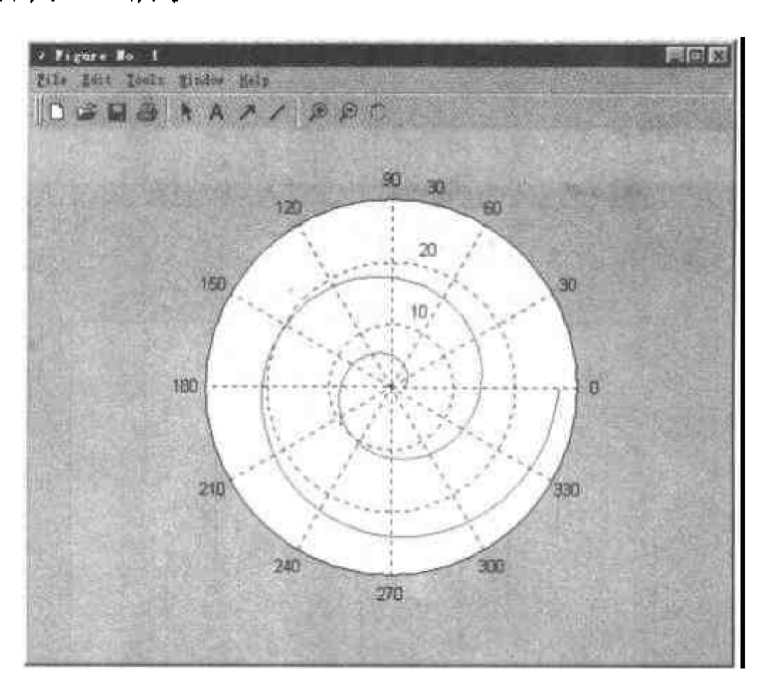


图 3-50 极坐标下的渐开线

3.6.2 柱坐标系和球坐标系下绘制图形

在 MATLAB 中没有在柱坐标和球坐标下直接绘制数据图形的命令，但 pol2cart 和 sph2cart 命令能够将柱坐标值和球坐标值转化为直角坐标系下的坐标值，然后即可在直角坐标下绘制数据图形。

1. pol2cart 命令函数

pol2cart 命令用于将柱坐标转化为直角坐标，其调用格式为：

[*X*,*Y*] = pol2cart(*TH*,*R*)

[*X*,*Y*,*Z*] = pol2cart(*TH*,*R*,*Z*)

格式中的参数 *TH* 表示极坐标系下的角度向量或矩阵，*R* 是极半径向量或矩阵。矩阵 *TH*、*R* 和 *Z* 必须大小相等或成比例（*Z* 是极坐标下的高度向量或矩阵）。输出值 *X*、*Y*、*Z* 为直角坐标系下的坐标向量或矩阵。

2. sph2cart 命令函数

sph2cart 命令用于将球坐标转化直角坐标，其调用格式为：

[*X*,*Y*,*Z*] = sph2cart(TH,PHI,*R*)

格式中的参数 TH 为球坐标系下的方位角（Azimuth），PH*I* 是球坐标系下的俯视角（Elevation），*R* 是球半径。向量或矩阵 *TH*、*PHI* 和 *R* 的大小必须相等或成比例。输出值 *X*、*Y*、*Z* 为直角坐标系下的坐标向量或矩阵。

下面通过例 3-46 来说明它们的具体用法。

【例 3-46】函数 pol2cart 和 sph2cart 的用法。

```
theta=0:pi/20:6*pi;
rho=sin(theta);
[t,r]=meshgrid(theta,rho);
z=r.*t;
phi=theta.^2-theta;
 [t1,p1]=meshgrid(theta,phi);
r1=p1.*t1;
 [X,Y,Z]=pol2cart(t,r,z);
mesh(X,Y,Z)
figure(2)                    %绘制球坐标转化直角坐标时的图形
[X1,Y1,Z1]=sph2cart(t1,p1,r1);
mesh(X1,Y1,Z1)
```

其执行结果如图 3-51、图 3-52 所示。

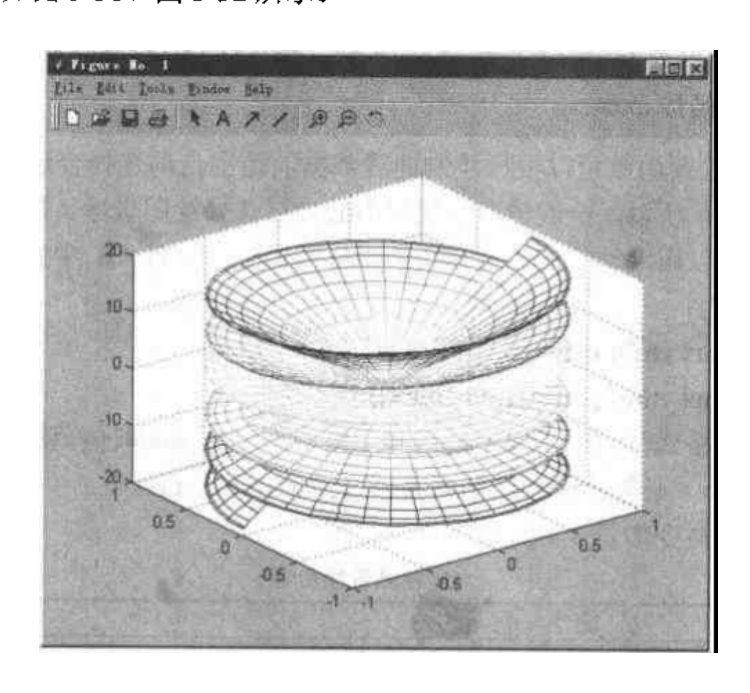


图 3-51 柱坐标绘图

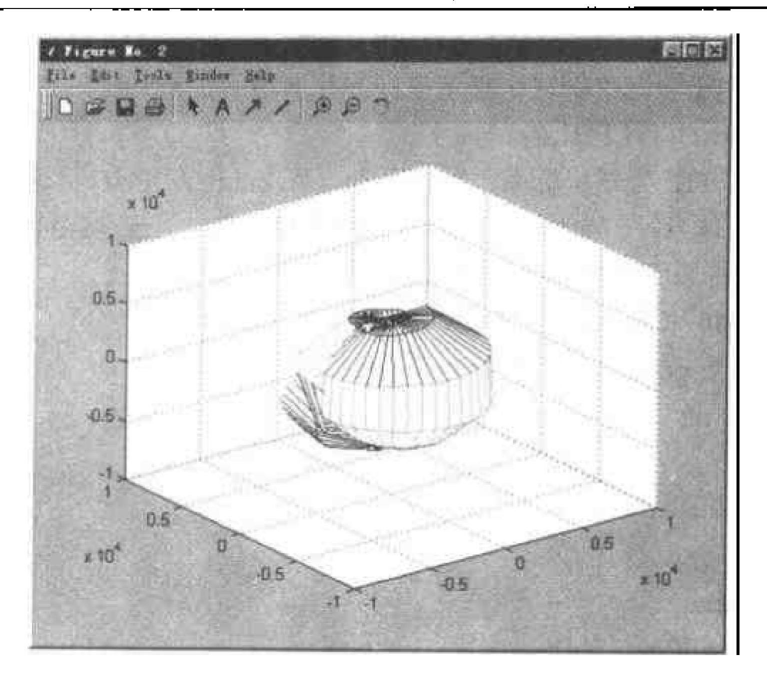


图 3-52 球坐标绘图

3.7 坐标轴的控制和图形标注

许多时候，用户需要对坐标轴进行调整，并对图形的内容进行标注，MATLAB 6.0 提供了特定的函数来完成这方面的工作。

3.7.1 坐标轴控制函数 axis

MATLAB 的绘图函数可以根据绘制曲线数据的范围自动选择合适的坐标系，使曲线尽可能清楚地显示，所以，一般情况下用户不必自己选择绘图坐标。如果觉得自动选择的坐标不合适，则可以用手动的方式选择新的坐标系。在 MATLAB 中完成此功能的函数是 axis，其调用格式为：

axis([*x*min *x*max *y*min *y*max])

axis([*x*min *x*max *y*min *y*max *z*min *z*max])

以上两种命令格式分别用于设定二维和三维图形坐标轴的取值范围，并且坐标的最小值（*x*min,*y*min,*z*min）必须小于相应的最大值（*z*max,*y*max,*z*max）。

坐标轴的调整命令如表 3-4 所示。

表 3-4 坐标轴调整命令

命令格式	功 能
axis auto	将坐标轴的取值设为默认值
axis manual	冻结当前的坐标轴，以保持坐标轴取值范围不变。即使用 hold 命令，也保持不变
axis tight	将在三个方向上的纵高比设为同一个值
axis fill	将坐标轴的取值范围分别设为绘图所用数据在相应方向上的最大、最小值

续表

命令格式	功　能
axis ij	将二维图形的坐标原点设置在图形的左上角，i 轴垂直向下，j 轴水平向右
axis xy	把二维图形的坐标轴恢复为系统默认状态
axis equal	使坐标轴在三个方向上的刻度增量相同
axis image	与 axis on 基本相同，在 plot 中，将起 axis tight 的作用
axis square	使坐标轴在三个方向上的长度相等
axis normal	撤消命令 axis equal 和 axis vis3d 的操作
axis vis3d	冻结坐标系的状态，以便进行旋转
axis off	使坐标轴消隐
axis on	恢复消隐的坐标轴

3.7.2　图形标注

MATLAB 可以在画出的图形上加各种标注及文字说明，以增强图形的表现力。在中文版 Windows 环境下，还可以用汉字进行标注。在此介绍其基本的标注命令，有关这方面的高级操作，将在图形句柄部分说明。

1. 坐标轴和图形标题标注

在 MATLAB 中，标注坐标轴 x、y 和 z 的相应命令函数为 xlabel、ylabel 和 zlabel。现以函数 xlabel 为例来说明其用法。

xlabel('text')

xlabel('text','Property1',PropertyValue1,'Property2',PropertyValue2,...)

H = xlabel(...)

其中，'text'是要添加的标注文本。'Property'是该文本的属性，'PropertyValue1'是相应的属性值，定义所用字体、大小以及标注角度等。该命令把文本'text'按照指定的格式添加到 x 轴下方。

在 MATLAB 中，给图形加标题的函数为 title，其调用格式与 xlabel 相似：

title('text')

title('text','Property1',PropertyValue1,'Property2',PropertyValue2,...)

H = title(...)

其中，'text'是用添加的标注文本：该命令与 xlabel 的区别只是 title 函数把文本'text'加到图形的上方。

在 MATLAB 5.3 以上版本中，用户也可以在图形窗口中通过快捷按钮在图像的任意位置添加字符或汉字。

MATLAB 5.x 以上版本中涉及图形字符串指令（如 xlabel、ylabel、zlabel、title 和 legend 等）中的设置属性很多，例如“FontWeight”（字体粗细）、“FontAngle”（字体角度）以及“FontSize”（字体大小）等，将在后面的图形属性中详细介绍。

另外，MATLAB 的文本字符串中有许多特征字符串来表示 TeX 格式的字符。使用特征字符串可以把很多数学公式或工程中的 TeX 符号标注到图形上，这样就大大增加了标注的灵活性。MATLAB 中的特征字符串及其对应的 TeX 符号如表 3-5 和表 3-6 所示。

表 3-5 特征字符串中的希腊字母

特殊字符串	符号	特殊字符串	符号	特殊字符串	符号
\alpha	α	\vartheta		\Pi	Π
\beta	β	\varsigma	ζ	\rho	ρ
\gamma	γ	\psi	ψ	\sigma	σ
\Gamma	Γ	\Lambda	Λ	\Sigma	Σ
\delta	δ	\mu	μ	\tau	τ
\Delta	Λ	\Nu	ν	\upsilon	υ
\epsilon	ε	\xi	ξ	\Upsilon	T
\zeta	ζ	\Xi	E	\phi	ф
\ata	η	\pi	π	\Phi	Φ
\theta	θ	\kappa	κ	\omega	ω
\Theta	Θ	\lambda	λ	\Omega	Ω
\iota	ι	\varpi	Λ	\Psi	Ψ

表 3-6 特征字符串中的其他字符

特征字符串	符号	特征字符串	符号	特征字符串	符号
\approx	≈	\partial	∂	\subset	⊂
\cong	≌	\exists	∃	\subseteq	⊆
\div	÷	\forall	∀	\supset	⊃
\equiv	≡	\in	∈	\supseteq	⊇
\geq	⩾	\infty	∞	\Im	ℑ
\leq	⩽	\perp	⊥	\Re	ℜ
\neq	≠	\prime	'	\downarrow	↓
\pm	±	\ cdot	•	\leftarrow	←
\propto	∝	\ldots	…	\leftrightarrow	↔
\sim	~	\cap	∩	\rightarrow	→
\times	×	\cup	∪	\uparrow	↑
\oplus	⊕	\otimes	⊗	\circ	∘
\nslash	ø	\int	∫	\bullet	•
\copyright	©				

注意：特征字符串区分大小写。

2. 图形标注

在 MATLAB 中，函数 text 和 gtext 两个命令专门用于标注所绘制的具体图形，可以在图形窗口的任何位置加入文本字符串。函数 text 的调用格式为：

text(*X*,*Y*,'string')

text(*X*,*Y*,*Z*,'string')

前者用于二维图形的标注，后者用于三维图形的标注。其中，*X*、*Y*、*Z* 表示标注字符串所在的位置。'string'是要添加的字符串，该字符串也可以是由“\”引导的特征字符串表示的特殊符号。当 *X*、*Y*、*Z* 为向量时，将在这三个向量决定的每一点上标注'string'，如果'string'也是与 *X*、*Y*、*Z* 同维的字符串向量，则在每一点上标注相应的字符串。

在 MATLAB 中，函数 gtext 适用于绘制图形标注的命令，它是用鼠标或键盘单击标注位置。其调用格式如下。

gtext('string')

gtext(...,'PropertyName',PropertyValue,...)

当 MATLAB 执行该指令时，系统当前的图形窗口被激活，鼠标在此窗口上将显示为一个小十字形，提示选取标注位置，可以在选定位置处单击鼠标或键盘进行确定。第二种格式是设置文本属性，多项文本属性可有一行简单的命令进行表述。例如：

```
gtext({'This is the first line','This is the second line'})
gtext({'First line','Second line'},'FontName','Times','Fontsize',12)
```

3. 图例的标注

当一幅图中出现多种曲线时，结合绘制时的不同线型和颜色等特点，用户可以用 legend 命令对不同的图例进行说明。其调用格式为：

legend(string1,string2,string3, ...)

legend(string1,string2,string3,...,Pos)

第一种调用格式：只要指定标注字符串该函数就会按顺序把字符串添加到相应的曲线线型符号之后。

注意：在 MATLAB 中还可以方便地对图例进行调整。用鼠标左键点住图例拖动即可移动图例到需要的位置；用鼠标左键双击图例中的某个字符串即可对该字符串进行编辑。

第二种调用格式：在函数 legend 中加入一个参数也可以对图例的位置作出设置和调整。调用格式中“Pos”的取值和含义如下：

- 0 =自动把图例置于最佳位置（和图中曲线重复最少）；
- 1 =置于图形窗口的右上角（缺省值）；
- 2 =置于图形窗口的左上角；
- 3 =置于图形窗口的左下角；
- 4 =置于图形窗口的右下角；
- -1 =置于图形窗口的右侧（外部）。

4. 控制分格线

在 MATLAB 中分格线的控制命令对二维和三维图形都适用，并且使用频率较高，使用方法简便。控制分格线命令有三种用法，分别为：

- grid on　打开分格线控制开关，以后绘制的图形都带有分格线；
- grid off　关闭分格线控制开关，以后绘制的图形都不带分格线；
- grid　用于是否实现分格线绘制切换。

下面将根据以上讲述的内容，举一个综合的例子来具体体现上面几个命令。

【例 3-47】绘制一图形，用函数 xlabel、title 和 legend 命令进行标注。

```
t=0:0.1:4*pi;
y=sin(t);
y1=cos(t);
plot(t,y,':',t,y1,'r*')
xlabel('x 轴  (0--4\pi)','fontsize',12,'fontweight','bold')
ylabel('y 轴','fontsize',12,'fontweigbt','bold')
title('绘制正弦波和余弦波      Fos=1','fontsize',10,'fontweight','bold','fontangle','italic')
```

```
text(pi,0,'\leftarrowsin(\pi)=0')
text(pi,-1,'\leftarrowcos(\pi)=-1')
text(pi/2,0.9,['\uparrowsin(\pi/2)=',num2str(sin(pi/2))])
text(0,-0.6,['绘图日期：',date])
text(0,-0.8,['MATLAB 版本：',version])
legend('正弦波','余弦波')
figure(2)
title('绘制正弦波和余弦波  Pos=0','fontsize',10,'fontweight','bold','fontangle','italic')
legend('正弦波','余弦波',0)
grid on
figure(3)
title('绘制正弦波和余弦波 Pos=-1','fontsize',10,'fontweight','bold','fontangle','italic')
text(7*pi/2,0,'\rightarrowcos(\pi*7/2)=0')
legend('正弦波','余弦波',-1)
grid off
```

其执行结果如图 3-53、图 3-54、图 3-55 所示。

注意：在命令 text(pi/2,0.9,['\uparrowsin(\pi/2)=',num2str(sin(pi/2))])中，sin(pi/2)是一个实数，而不是字符串，此时不能和前面的字符串连接，必须先用 num2str 函数把它转化为字符串。字符串用中括号括起来的部分表示它们为一整体；在命令 text(0,-0.6,['绘图日期: ',date])中，date 函数给出当前的日期，version 函数给出使用的计算机类型，它们返回的都是字符串，不用进行类型转换，可以和其他字符串直接连接。

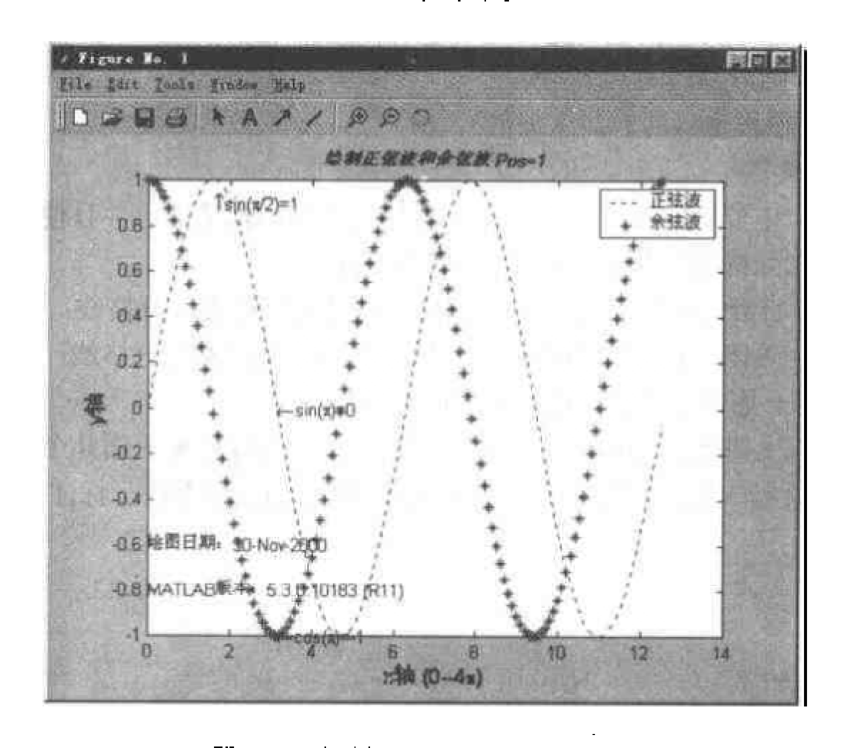


图 3-53 绘制正弦波和余弦波 Pos=1

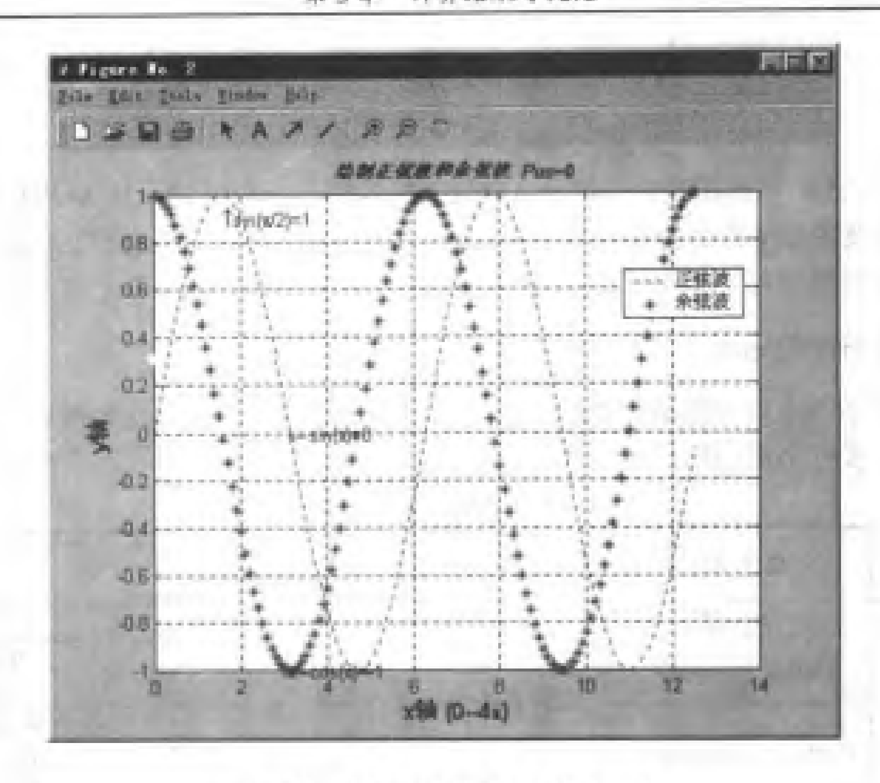


图 3-54　绘制正弦波和余弦波 Pos=0

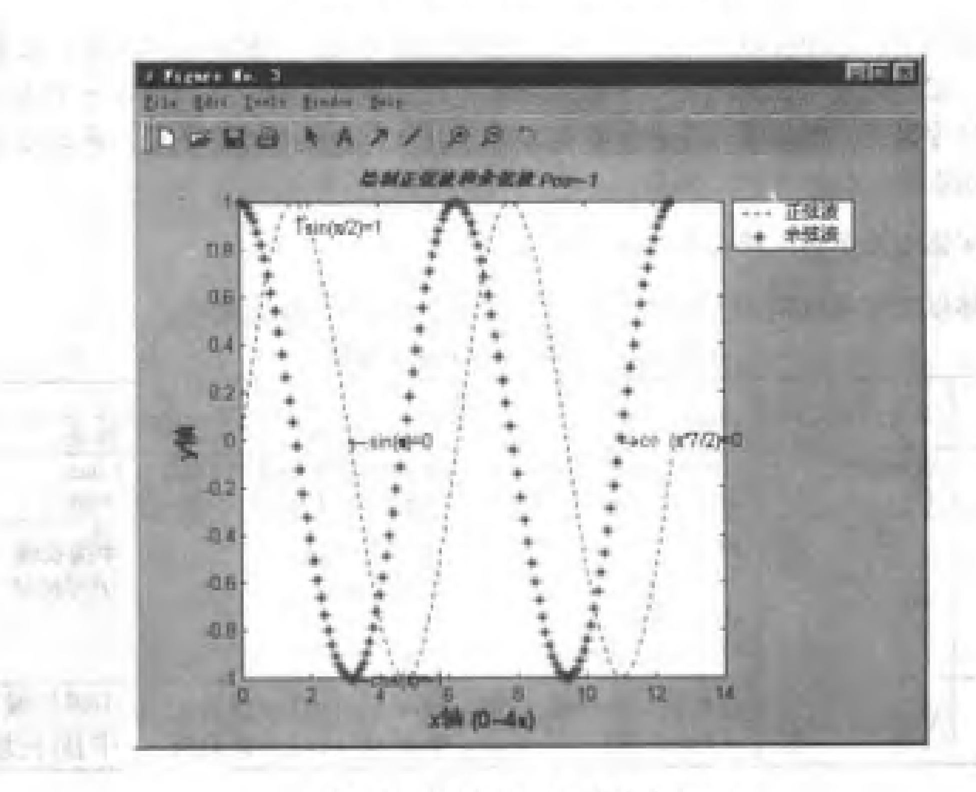


图 3-55　绘制正弦波和余弦波 Pos=-1

3.8 MATLAB 的图形标注精细命令

3.7 节所讲述的图形标注命令都是 MATLAB 的简捷命令形式，从 MATLAB 5.x 版本开始，所有涉及图形字符串的标注命令（如 title、xlabel、ylabel、zlabel、legend、text、gtext 等）都能对字符串进行更精细的控制。

3.8.1 多行字符串的标注

标注多行字符串可以用单元数组，也可以用多行字符串数组。比较而言，单元数组更加灵活方便。多行字符的标注如表 3-7 所示。

表 3-7 多行字符串标注的规则

	命令格式	string 的取值	举 例	
			示例命令格式	效果
单行	'string'	任何合法的字符	'中国'	中国
多行	{'string1','string2'}		{'中国 长城','China'}	中国 长城 China
	['string1';'string2']		[' 中 国 长 城 ';'China';'Line ']	中国 长城 China Line

注意：当用单元数组存放多行字符串时，每行字符串为一个单元数组元素，元素间的分隔用逗号、空格或分号。当用字符串数组存放多行字符时，每个字符行占一个数组行，数组元素用分号隔开。数组每行字符个数必须相等，如第一行为 5 个字符，第三行为 4 个字符，就必须在第三行中加一个空格。

3.8.2 标注字体以及字体风格和大小的设置

标注字体以及字体风格和大小的设置，要符合表 3-8 中的设置规则。

表 3-8 标注字体、风格和大小的设置规则

类别	命令格式	string 的取值	举 例	
			示例命令格式	效果
字体	\fontname {rg}	arial; courier; roman 宋体；隶书；黑体…	'\fontname {courier} Line' '\fontname {隶书} 中国'	Line 中国
风格	\arg	Bf （黑 体） it （斜体一） sl （斜体二） rm （正 体）	'\bf 中国长城' '\it 中国长城' '	**中国长城** *中国长城*
大小	\fontsize {arg}	正整数。缺省值为 10（Points 磅）	'\fontsize {10} 中国长城' '\fontsize {12} 中国长城'	中国长城 中国长城

说明：

- Windows 字库中所有的字体，都可以通过设置字体名称实现调用；
- 中文字体都允许调用；

- Point（磅）=1/72Inch=0.35mm。

3.8.3 上下标的设置

文本标注时常遇到上下标的问题，其控制命令如表 3-9 所示。

表 3-9 文本上下标标注的设置规则

	命令格式	arg 的取值	举例	
			示例命令格式	效果
上标	^{arg}	任何合法字体	'\ite^{-t}sint'	$e^{-t}\sin t$
下标	_{arg}	任何合法字体	'\x~{\chi}_{\alpha}^{2}(3)'	$x \sim \chi_{\alpha}^{2}(3)$

3.9 MATLAB 6.0 中的新增函数

在 MATLAB 6.0 中关于三维可视化方面新增的函数如表 3-10 所示，此类函数大多用于较为复杂的图形图像处理，属于对较高层次的要求。鉴于多数用户的使用范围不是十分深入，下面将以函数 streamribbon、streamslice 和 streamtube 为例，通过例 3-48、例 3-49 和例 3-50 来说明函数在模拟风流动时的使用，其余函数的用法大致类似，在此不予详细叙述，有特殊需要的用户可通过 MATLAB 6.0 的文件来做进一步的学习和应用。

表 3-10 MATLAB 6.0 的三维可视化新增函数

函数名称	功能
coneplot	创建三维圆锥体
contourslice	在立体切面上绘制等高线
curl	计算垂直于三维向量空间流动方向的斜率和曲率
divergence	计算三维向量空间的发散度
interpstreamspeed	内插速度流线顶点
isocolors	计算等面顶点的颜色
isosurface	提取等面的几何数据
streamparticles	显示流动粒子
streamribbon	显示流动带
streamslice	显示流线切片图
streamtube	显示流线管图
volumebounds	返回坐标和颜色的界限

【例 3-48】利用流动带函数显示风数据的流动。

（1）设置输入数据，并调用函数 streamribbon 显示流动带。

```
load wind
[sx sy sz] = meshgrid(80,20:10:50,0:5:15);
daspect([1 1 1])
streamribbon(x,y,z,u,v,w,sx,sy,sz);
```

（2）定义视图和光源。

```
axis tight;axis on;
```

```
shading flat;
view(3);
camlight; lighting gouraud
```

程序的运行结果如图 3-56 所示。

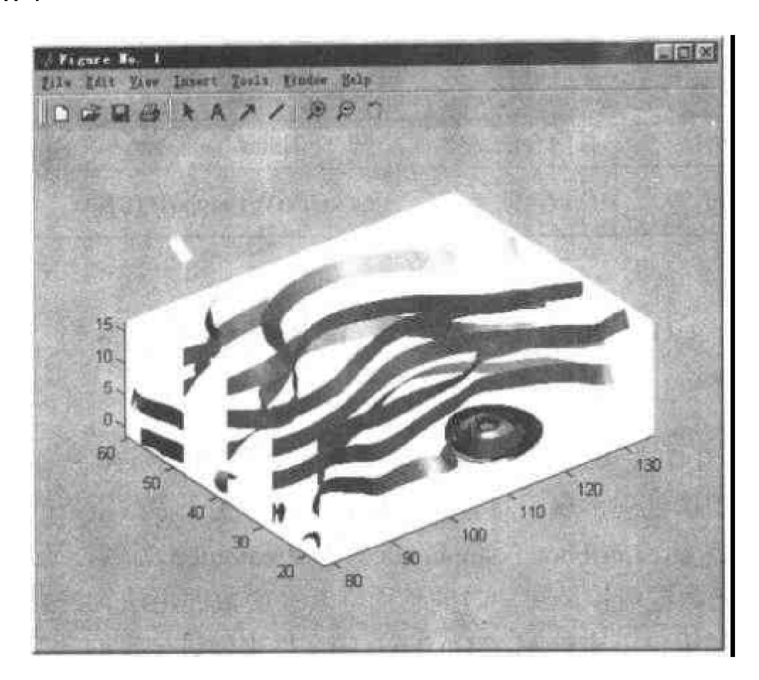


图 3-56 流动带函数调用显示图像

【例 3-49】利用函数 streamslice 显示风流动的切片图。

```
load wind
daspect([1 1 1])
streamslice(x,y,z,u,v,w,[],[],[5])   %设置 z=5
axis tight
```

程序的运行结果如图 3-57 所示。

【例 3-50】利用函数 streamtube 显示风流动的流动管状图。

（1）设置输入数据，并调用函数 streamribbon 显示流动带。

```
load wind
[sx sy sz] = meshgrid(80,20:10:50,0:5:15);
daspect([1 1 1])
streamtube(x,y,z,u,v,w,sx,sy,sz);
```

（2）定义视图和光源。

```
view(3)
axis tight
shading interp;
camlight; lighting gouraud
```

程序的运行结果如图 3-58 所示。

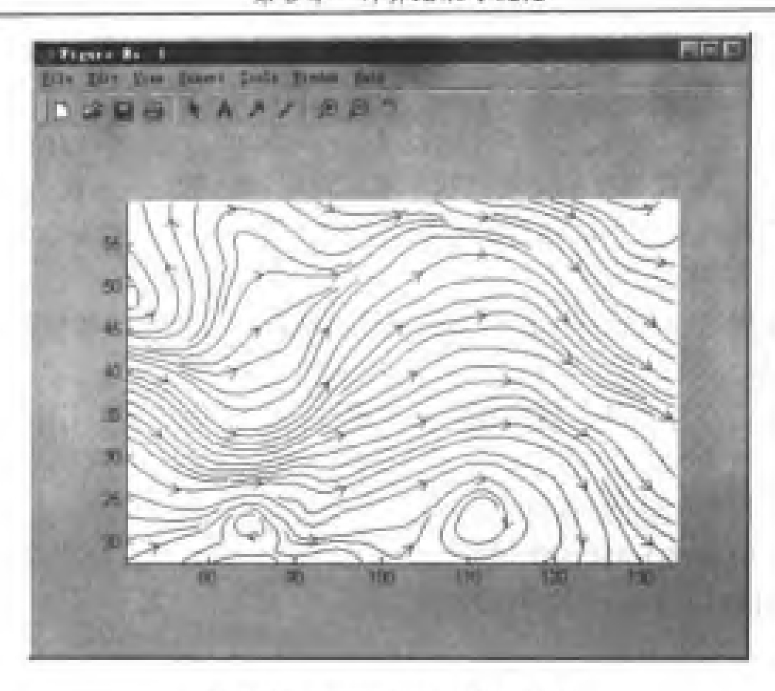


图 3-57　streamslice 绘制的图像

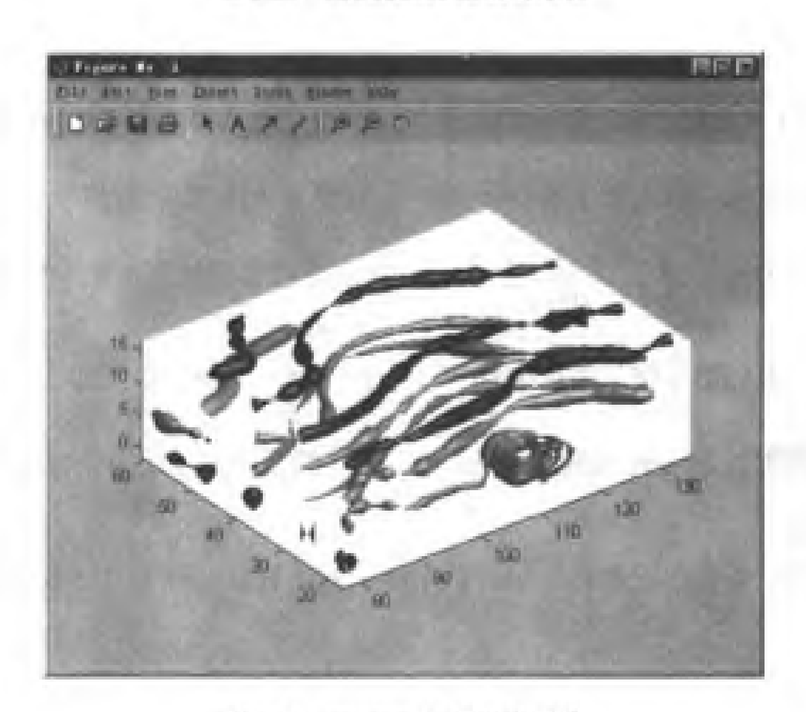


图 3-58　streamtube 绘制的图像

3.10　小　　结

MATLAB 具备强大的数据可视化功能，可以将复杂的数据用二维、三维乃至四维图形表现出来。通过对数据得出图形的线型、立面、色彩、渲染、光线以及视角等属性的处

理，将计算数据的特性表现得淋漓尽致。通过对本章的学习，应掌握如下内容：

（1）二维平面图形与三维绘图方面，掌握如何创建与控制图形输出窗口以及对图形窗口的操作；熟练掌握二维和三维基本的绘图命令、线面着色以及线型控制；三维图形的照明和材质处理及其可视效果的控制；初步掌握轴面和球面的三维表达方法。

（2）学会用四维表现图展示复杂数据的特性，掌握分别用色彩、切片图和等位线图来对图形进行四维表现。

（3）初步掌握用一些特殊的图形来表现特殊数据的性质，如面积图、直方图、饼图、柱状图和阶梯图等。

（4）除了平面坐标系外，要掌握如何在极坐标、柱坐标和球坐标系下绘制二维图形和三维图形。

（5）熟练掌握坐标轴的控制和图形标注命令及其用法。

（6）初步了解 MATLAB 的图形标注精细命令，使根据数据绘制的图形更具表现力。

（7）了解 MATLAB 6.0 中的新增函数。

习　　题

1．按照$\Delta x = 0.1$的步长间隔绘制函数$y = xe^{-x}$在$0 \le x \le 1$时的曲线。

2．用图形表示离散函数$y = |(n-6)|^{-1}$，其中n为[0,12]的自然数。

3．分别采用$\Delta t = \frac{1}{10\pi}$、$\frac{1}{100\pi}$的步长，绘制连续调制波形$y = \sin(t)\sin(9t)$的图像。

4．给出一系列的a值，采用函数$\frac{x^2}{a^2} + \frac{y^2}{25 - a^2} = 1$画一组椭圆。

5．用曲面图命令 surf 表现函数$z = x^2 + y^2$的图像，并用火柴杆命令 stem3 表示网格点上的函数值。

6．绘制颜色为蓝色，数据点用五角星标识的函数$y=xe^{\sin x}$在(0,5)上的虚线图。

第 4 章　MATLAB 程序设计基本知识

MATLAB 作为一种高级应用软件，除了命令行操作的直接交互方式以外，还有自己的编程语言。为了充分发挥和体现 MATLAB 的功能，必须掌握 MATLAB 的程序设计。本章将详细介绍MATLAB 程序设计的基本知识，通过本章的学习，可以掌握关于MATLAB程序设计的方法，增强设计应用程序的能力。

MATLAB 不仅是一个功能强大的工具软件，更是一种高效的编程语言。MATLAB 的编程效率比常用的 BASIC、C、FORTRAN 和 PASCAL 等语言要高得多，而且容易维护。MATLAB 软件即 MATLAB 语言的编程环境，M 文件也就是用 MATLAB 语言编写的程序代码文件。

本章将从语言的角度介绍编写 MATLAB 基本程序的规则和方法。

4.1　MATLAB 的变量与表达式

4.1.1　MATLAB 的变量与类型

1. *变量命名规则*

在 MATLAB 中，对变量（包括函数）命名时应遵循以下规则：

- 变量名和函数名对字母的大小写敏感，即 MATLAB 区分字母的大小写；
- 变量名的第一个字符必须是英文字母，最多可包括 31 个字符；
- 变量名可由英文字母、数字和下划线混合组成；
- 变量名中不得包含空格（Backspace）和标点，但可以有下连字符。如变量名 my_var_330 就是合法的。

2. *局部变量和全局变量*

通常每个函数体内都有自己定义的变量，不能从其他函数和 MATLAB 工作空间访问这些变量，这就是局部变量。如果要使某个变量在几个函数及 MATLAB 函数中都能使用，它就是全局变量。

全局变量名应尽可能大写，并用“global”声明。如果要在几个函数和 MATLAB 的工作空间中都能访问一个全局变量，则需在每个函数和 MATLAB 工作空间中都声明该变量是全局变量。全局变量要在函数体的变量赋值语句之前说明，整个函数以及所有对函数的递归调用都可以利用全局变量。

注意： 在实际编程中，应尽量避免使用全局变量，因为全局变量的值一旦改动，则在其他包括该变量的函数中都将改变，这样有可能会出现不可预见的情况。

3. 永久变量

上面提到定义变量时，有些 MATLAB 的保留字符不能用，其中有一部分就是 MATLAB 的永久变量，也称为预定义变量（Predefined Variable）。每当 MATLAB 启动时，系统自动定义变量，驻留于内存中。它们不会被命令 clear 清除（永久变量的名称就源于此）。系统也可以为这些永久变量赋值，但所赋的值可以用 clear 命令清除，从而恢复系统预定义的值（预定义变量的名字就反映这个意思）。who 命令看不到永久变量（除非是适用的 MATLAB 3.0 版）。MATLAB 的永久变量如表 4-1 所示。

注意：用户在编写指令和程序时，尽可能不对表 4-1 中所列永久变量名重新赋值，以免产生混淆。

表 4-1 MATLAB 的永久变量

永久变量名	含义	永久变量名	含义
ans	计算结果的缺省变量名	NaN 或 nan	非数（Not a Nmber），如 0/0
Eps	容差变量，定义为 1.0 到最近浮点的距离	nargin	函数输入总量数目
flops	浮点运算次数	nargout	函数输出总量数目
Inf 或 inf	无穷大，定义为 1/0	realmax	最大的浮点数
pi	圆周率 π	realmin	最小的浮点数
i 或 *j*	虚数单位为 $i=\sqrt{-1}$，$j=\sqrt{-1}$		

【例 4-1】无穷大的使用和作用。

```
y=1/0        %无穷大的使用
1/y          %无穷大的作用和 ans 变量的使用
```

结果为：

```
Warning: Divide hy zero.
y =
    Inf
ans =
    0
```

注意：在 MATLAB 中，像 1/0 这样的操作不会引起程序执行中断，只是在给出警告信息的同时，用一个永久变量 inf 来表示，而且这个变量和其他变量一样，可在各种运算中发挥巨大的作用。

对于 *y*(inf)同样可以当作一个变量来使用，此特点在编程中有很大作用。

4.1.2 MATLAB 基本表达式

MATLAB 采用的是表达式语言，用户输入的语句由 MATLAB 系统解释运行。用户可以在 MATLAB 的命令窗口中键入命令，也可以在编辑器内编写应用程序，MATLAB 软件对此命令或程序中各条语句进行翻译，然后在 MATLAB 环境下对它进行处理，最后返回运算结果。MATLAB 语句由表达式和变量组成，有两种最常见的语句形式：

表达式

变量=表达式

MATLAB 书写表达式的规则与“手写算式”几乎相同，具体规定为：

- 表达式由变量名、运算符、数字和函数名组成。它在 MATLAB 中占有很重要的位置，几乎所有的运算都必须借助表达式来进行；
- 表达式将按常规的优先级从左至右执行运算；
- 优先级的规定是指数运算级别最高，乘除运算次之，加减运算级别最低；
- 括号可以改变运算顺序；
- 书写表达式时，赋值符“=”和运算符两侧允许有空格，以增加可读性。但在复数或符号表达式中要尽量避免，以防出错；
- 表达式的末尾可加上“；”，也可以不加。有“；”时，MATLAB 系统不显示计算结果，而是直接把数值赋给变量，如果没有用变量就无法看到结果；没有“；”时，MATLAB 系统将会在该条语句的下面直接显示运算结果。

在 MATLAB 语句的第一种形式中，表达式运算后产生的结果如果是矩阵或其他数值类型，MATLAB 系统将会自动赋给变量名为 ans，并显示在屏幕上。ans 是一个默认的永久变量，它会在以后类似的操作中被自动覆盖掉，所以重要的结果一定要记录下来，也就是使用第二种形式。

在 MATLAB 语句的第二种形式中，“=”右边的表达式计算后产生的结果，MATLAB 系统会自动赋给“=”左边的变量，然后存入内存中，并显示在屏幕上。

MATLAB 语言与 C 语言不同，它允许一次返回多个结果，这时“=”左边是用“[ ]”括起来的变量列表，例如命令函数：

[*X*，*Y*，*Z*]=peaks；

它返回了坐标列向量 *X*，*Y*，*Z* 的值。

【例 4-2】创建一个表达式。

```
x=(3*4^2+exp(sin(45)))/6
x =
    8.3903
```

4.2 字符串数组、单元数组和结构数组

4.2.1 MATLAB 的数据结构

MATLAB 是一种面向数组（Array）的编程语言，其数据类型的最大特点是每一种类型都以数组为基础，从数组中派生出来的，事实上，MATLAB 把每种类型的数据都作为数组来处理。在 MATLAB 6.0 中，有 6 种基本的数据类型，即：char（字符）、double（双精度数值）、sparse（稀疏数据）、storage（存储型）、cell（单元数组）和 struct（结构）。数据类型间的关系如图 4-1 所示。

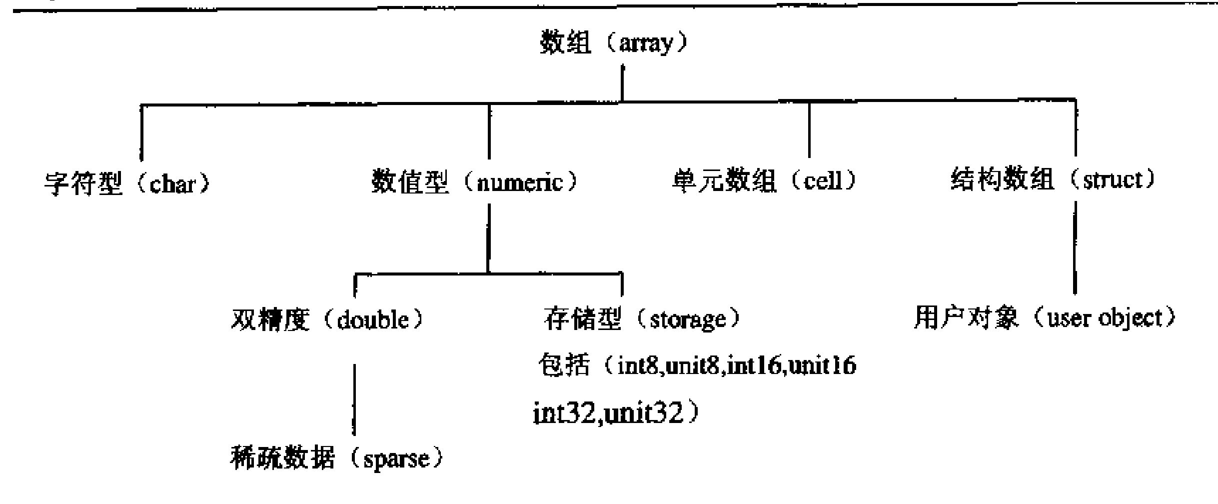


图 4-1 MATLAB 6.0 的数据结构

说明：

- 在图 4-1 中，存储型是一个虚拟数据类型，这是 MATLAB 5.3 版以后新增的定义，它包括 int8（8 位整型）、uint8（无符号 8 位整型）、int16（16 位整型）、uint16（无符号 16 位整型）、int32（32 位整型）和 uint32（无符号 32 位整型）。
- 最常用的数据类型只有双精度型和字符型，所有 MATLAB 计算都把数据当作双精度型处理。
- 其他数据类型只在一些特殊条件下使用。

例如，无特号 8 位整型一般用于储存图像数据；稀疏数据一般用于处理稀疏矩阵；单元他组和他构他组一般用在大型程序中。

- 存他型他组一般只用于内存的有他储存，可对这些类型的数组进行操作，但不能进行任何数学运算，否则必须用 double 函数把它转换为双精度类型。

MATLAB 的字符串他组（Character String Array）、单元数组（Cell Array）和结构数组（Structure Array）之间以及与数值数组（Numeric Array）之间的基本差别如表 4-2 所示。

表 4-2 MATLAB 中 4 种数据类型的比较

数组类型	基本成分	成分的含义	基本成分所占用字节数
数值数组	元素	双精度实数或复数标量	8 或 16
字符串数组	元素	字符	2
单元数组	单元	可以存放任何类型、任何大小的数据	不定
结构数组	结构	只有挂接在结构上的“域”才能存放数据。数据可以是任意类型、任意大小	不定

4.2.2 MATLAB 字特串数组

对于 MATLAB 编程来说，字符处理必不可少。对于字符串的使用应特合以下规定。

- 所有字符串都用单引号括起来。
- 字符串中的每个字符都是字符串变量（矩阵或向量）中的一个元素。
- 字特串中的字符以 ASCII 码形式储存并区分大小。用函数 abs 可以看到字符的 ASCII 码。

MATLAB有强大的字符处理功能，其字符串处理函数如表4-3所示。

表4-3 字符串处理函数

函数名称	功能和含义	函数名称	功能和含义
abs	将字符串转换为ASCII码值	texlabel	用特征字符串产生TeX格式的符号
eval	解释执行字符串	char	建立或转换字符串数组
deblank	删除字符串末尾的空格	int2str	整数转换为字符串
findstr	从一个字符串中查找另一个字符串	mat2str	矩阵转换为字符串
lower	将字符串转换为小写	num2str	数字转换为字符串
upper	将字符串转换为大写	sprintf	将带格式的数字转换为字符串
strcat	将字符串水平连接	sscanf	将字符串转换为带格式的数字
strcmp	比较字符串	str2double	字符串转换为双精度数
strcmpi	忽略大小写比较字符串	str2num	字符串转化为数字
strncmp	比较字符串的前n个字符	bin2dec	二进制数转化为十进制数
strmatch	查找匹配的字符串	dec2bin	十进制数转化为二进制数
strrep	替换字符串	dec2hex	十进制数转换为十六进制数
strjust	对齐字符数组（左对齐、右对齐、居中）	hex2dec	十六进制数转换为十进制数
strtok	返回字符串中第一个分隔符（包括空格、Enter和Tab键）前的部分	hex2num	十六进制数转换为双精度数
strvcat	垂直连接字符串可建立多行字符串	symvar	确定字符串中的符号变量（除常量和函数外的内容）

下面介绍几种常用的字符串操作。

1. 字符串数组的建立

● 直接赋值法建立字符串数组

建立字符串可通过直接赋值，先把待建的字符放在单引号中，然后直接赋值给变量。如：

```
a='This is a book'
a =
    This is a book
```

● 建立中文字符串数组

创建中文字符串时，字符外边的单引号对必须在英文状态下输入。与英文字符一样，每个中文字符也占一个元素位置，但应注意此时的ASCII大于256。

如：A='中国 长城'

```
A =
    中国 长城
```

● 建立带单引号的字符串

当字符串中的字符包含有（英文）单引号时，每个单引号符用连续的两个单引号符表示。

```
    B='China''中国'''
B =
    China'中国'
```

● 有效字符串连成长字符串

字符串可以连接到一起组成更大的字符串。可以直接在中括号内用逗号连接，也可以通过函数 strcat 连接。如：

```
ab=[A,'',B,'.']                %第二个输入是表示输入空格字符串
ab =
    中国 长城 China'中国'.
strcat(A,B)                    %函数 strcat 连接，忽略原字符串结尾处的空格
ans =
    中国 长城 China'中国'
```

● 多行字符串数组的直接创建

在直接创建多行字符串数组时，要保证同一字符串数组的各行字符数相等，即保证各行等长。如果不等长，则用空格符来调节其长度，使它们彼此相等。如：

```
AB=['中国            ';'chang cheng']
AB =
    中国
    chang cheng
```

● 利用字符串操作函数创建字符串数组

在 MATLAB 中，有专用函数 char、str2mat 和 strvcat 创建多行字符串数组。这三个函数创建多行数组时，不必担心每行字符是否相等，它们总会按最长行设置第二维的长度，其他行的尾部用空格填充。它们的调用格式为：

S = char(T1,T2,T3,..)

S = str2mat(T1,T2,T3,..)

S = strvcat(T1,T2,T3,..)

上式中，*T*1，*T*2，*T*3,..是字符串，在输入时，*T*1，*T*2，*T*3…两边要加英文状态下的单引号。

● 利用转换函数建立字符串数组

在 MATLAB 中，把数值数组转化为字符串数组的常用函数为 int2str、num2str 和 mat2str。其调用格式可用命令 help int2tre、help num2str 和 help mat2str 来查看。从 MATLAB 5.x 版本以后，又增加了两个命令 char 和 double，函数命令 char 可把 ASCII 码数组转化为字符串数组；命令 double 可把其他任何数组转化为数值数组。

注意：中文字符能被命令 char 和 double 正确转化。

2. 字符串数组的元素标识

在一维字符串数组中，MATLAB 按从左向右的顺序用自然数数码（1，2，3 等）标识字符位置，其操作也用标识来进行。

4.2.3 MATLAB 单元数组

单元数组（Cell Array）是一种比较特殊的 MATLAB 数组，该数组的基本成分是单元，它的每个元素都是一个单元，单元中包含其他 MATLAB 数组。每个单元本身在数组中是平等的，它们只能以下标区分。单元内可以存度任何类型、任何大小的数组，而且同一单

元数组内各单元的内容可以不同。

同数值数组一样，单元数组的维数不受限制，可以是一维、二维或更高维。单元数组对单元的编址方法有单下标编址和全下标编址。

对于单元数组来说，单元和单元里的内容是两个不同的范畴。因此，寻访单元和寻访单元内容是两种不同的操作。以二维单元数组为例，*A*（2，3）是指 *A* 单元数组中的第二行第三列单元元素；而 *A*{2，3}是指 *A* 单元数组中的第二行第三列单元中所允许存或取的内容。两者的区别仅是用圆括号或是用花括号。

1. 单元数组的建立

在 MATLAB 中，有三种方法来建立单元数组。

- 利用赋值语句建立单元数组

赋值语句建立单元数组又有两种不同的方式：

◆ 用小括号括起单元的下标，在赋值语句的右侧用花括号括起单元的内容。如：

```
A(1,1)={'matlab'};              %单元（1，1），字符串
A(1,2)={6.0};                   %单元（1，2），标量，记录版本号
A(2,1)={'第阵'};                %单元（2，1），中文字符串
A(2,2)={[2 3 4; 23 1 4;5 4 8]}; %单元（2，2），矩阵
A                               %列出第阵A
A =                             %结果
      'matlab'      [          6]
      '矩阵'        [3x3 double]
```

◆ 用花括号括起单元的下标，在赋值语句的右侧直接指定单元的内容。如上面命令可写成：

```
A{1,1}='matlab';               %单元（1，1）中的内容
A{1,2}=6.0;                    %单元（1，2）中的内容
A{2,1}='矩阵';                 %单元（2，1）中的内容
A{2,2}=[2 3 4; 23 1 4;5 4 8]; %单元（2，2）中的内容
```

- 利用单元数起法（花括号）建立单元数组

也就是在花括号中直接赋值，单元与单元之间用逗号、空格或分号隔开，使用如下语句也可得出与上面相同的单元数组。

A={'matlab',6.0;'矩阵'，[2 3 4; 23 1 4;5 4 8]}或*A*={'matlab' 6.0;'第阵' [2 3 4; 23 1 4;5 4 8]}

上式中，6.0 后面的分号表示数组另起一行，不起用别的符号代替。

- 利用函数 cell 建立单元数组

cell 函数用来预分配指定大小的单元数版，其调用格式为：

◆ c=cell(*n*)　建立 $n \times n$ 的单元数组，单元是空矩阵；

◆ C=cell(*m*,*n*)或 cell([*m*,*n*])　建立 $m \times n$ 的单元数组，单元是空矩阵；

◆ c=cell(*m*,*n*,*p*,…)或 cell([*m n p* ...])　建立 $m \times n \times p \times \ldots$的单元数第，单元是空第阵；

◆ c=cell(size(*A*))　建立和 *A* 大小相同的单元数组。

2. 单元数组的访问和显示

在 MATLAB 中，单元数组的访问有两种方式：

● 用内容下标（花括号）访问单元内容

如上面的单元数组 *A*：

```
A{1,1}            %访问第(1,1)个单元中的内容
ans =
  matlab
```

● 用单元下标（小括号）访问单元子集

又如数组 *A*：

```
A(1,1)            %访问第(1,1)个单元
ans =
    'matlab'
b=A(1:2,1)        %把单元数组 A 中第一列的两个单元赋给 b，b 也成为单元数组
b =
    'matlab'
'矩阵'
```

单元数组中的内容一般以压缩的形式显示。要形象地查看单元数组中的内容，MATLAB 提供了函数 cellplot 命令，而函数 celldisp 命令只是显示单元数组全部或部分内容。函数 cellplot 的调用格式为：

H = cellplot(C,'legend')：式中第二个输入参数用于现实色彩图例，该命令用大白方格表示单元，用小方格表示所存的数组元素，色彩表示数据属性。如数组 *A* 用图形方式显示为：

```
cellplot(A,'legend')
```

结果如图 4-2 所示。

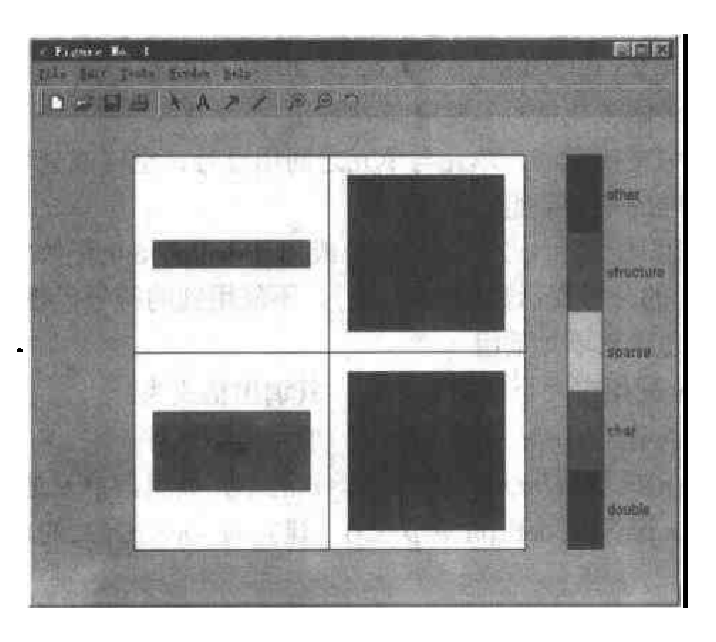


图 4-2　图形方式显示单元数组

3. 单元数组的操作

在 MATLAB 中，可对单元数组进行一系列操作。如：

- 把单元（注意，不是单元内容）设置为空来删除该单元数组中的单元；
- 用函数 reshape 命令来改变单元数组的大小，但要注意改变形状时数组中的元素个数；
- 单元数组也可像一般数组那样用方括号进行连接；
- 单元数组可代替用逗号或空格分隔的变量列表；
- 如果数组中的多个单元是用花括号和冒号一起引用的，MATLAB 则把每个单元的内容看作一个独立的变量。

4.2.4 MATLAB 结构数组

MATLAB 中的结构数组（Structure Array）与单元数组一样，能在一个数组存放各类数组。从一定意义上讲，结构数组组织数据的能力比单元数组更强、更富于变化。结构数组的基本成分是结构（Structure），数删中的每个结构相等，它们以下标区分。结构必须在划分“域”后才能使用。数据不能直接存放在结构上，而只能存放在域中。结构的域可以存放任何类型、任何大小的数组，而且不同结构的同名域中存放的内容可以不同。

1. 结构数组的建立和访问

- 直接建立结构数组

此种方法是直接建立结构和各个域，同时给各域赋值（也可以不赋值），结构和域之间用点连接。在访问结构数组的各个域时，也是“结构数组名.域名”的格式。当结构带有子域时，需完整地键入结构名、域名、子域名，才能显示域中的内容。结构数组的各个域可以按照其本身的数据类型进行相应的各种运算。下面我们通过一个例子来说明如何创建和访问域构数组。

【例 4-3】建立一桥梁混凝土试件强度的结构数组。

```
bridge.name='XX 大桥';                    %域名为 name，中文字符串，记录桥名
bridge.position='顶板';                   %域名为 position，中文字符串，记录试件位置
bridge.position.group='2 组';             %子域名为 group，字符串，记录组数
bridge.position.value=[28.4 29.4 30.1;26.8 29.0 28.8];  %子域名 value，记录试件强值
bridge                        %得到结构数组的内容如下
bridge =
        name: 'XX 大桥'
    position: [1x1 struct]
bridge.position               %显示域 position 中的内容
ans =
    group: '2 组'
    value: [2x3 double]
bridge.position.value         %显示域 value 中的内容
ans =
    28.4000    29.4000    30.1000
```

26.8000 29.0000 28.8000

用函数 structure 命令创建结构数组

除了上面所讲述的直接建立结构数组外，MATLAB 还有一个专门建立结构数组的函数 structue。该函数命令的调用格式为：

S = struct('field1',values1,'field2',values2,...)

用指定的域名和各个域的数据建立结构数组。

如果数组中包含多个结构，而且各个结构域中的数据不尽相同，则域的数据 values1、values2...必须是单元数组。建立的结构数组和单元数组的大小相同。如：

```
s = struct('type',{'big','little'},'color','red','x',{3 4})      %建立结构数组结果为：
s =
    1x2 struct array with fields:
    type
    color
    x
```

在 MATLAB 中，对结构数组域中内容的调取和设置可通过函数 getfield 和 setfield 进行；利用函数 rmfield 可以删除结构的域；利用函数 struct2cell 和 cell2struct 可以进行结构数组和单元数组之间的转换。这些命令的调用格式和使用，可参看 MATLAB 的帮助文件。

2. 数据的分配

MATLAB 中的函数 deal 可把输入数据分配给输出数据，其调用格式为：

[*A*,*B*,*C*,...] = deal(*X*,*Y*,*Z*,...)：等价于 *A*=*X*, *B*=*Y*, *C*=*Z*, ...

[*A*,*B*,*C*,...] = deaL(*X*)：等价于 *A*=*X*, *B*=*X*, *C*=*X*, ...

这个函数对于结构和单元数组很有用。下面举一个例子来说明：

```
sys = {rand(3) ones(3,1) eye(3) zeros(3,1)};                      %建立单元数组 sys
[a,b,c,d] = deal(sys{:});                %把 sys 中的每个“单元”分配给相对应的变量
a =
    0.9501    0.4860    0.4565
    0.2311    0.8913    0.0185
    0.6068    0.7621    0.8214
b =
    1
    1
    1
c =
    1    0    0
    0    1    0
    0    0    1
d =
    0
    0
```

0

注意：例中的 sys{:}和 sys 不同，sys 是将整个单元数组分配给每个变量。

4.3　MATLAB 运算符与操作符

在 MATLAB 中，一般的运算符和操作符构成运算最基本的操作指令，例如加、减、乘、除和乘方等运算，这些指令几乎在所有计算机语言中都有，并且大同小异。在 MATLAB 中，几乎所有的运算都是以矩阵为基本的运算单元，这与大多数计算机语言不同，也是 MATLAB 的显著特点。

4.3.1　运算符

在 MATLAB 中，最常见的运算符如表 4-4 所示。

表 4-4　MATLAB 的常用运算符

运算符	含义	运算符	含义	运算符	含义
+	加法	-	减法	^	乘方
*	乘法	/	右除	\	左除

注意：在矩阵运算中，左除和右除有一定的区别。

4.3.2　操作符

在 MATLAB 中，操作符在资料构造和运算中非常有用。各操作符的含义如表 4-5 所示。

表 4-5　MATLAB 的操作符

操作符	含义	操作符	含义	操作符	含义
:	冒号	...	连续号	'	转置号
(	括号	,	逗号	=	赋值符号
.	小数点	;	分号	==	等于号
..	源目录	%	注释号	>	大于号

表 4-5 中几个常用操作符解释如下：

- 冒号“：”　冒号在矩阵的构造和运算中极为实用，它可以用来产生向量；用作矩阵的下标；部分地选择矩阵元素；进行行循环操作等。
- 续号“...”　如果一个命令很长，一行容不下，则可以在一行的末尾加三个或更多的点，表示此行未完，而在下一行继续。
- 分号“；”　在方括号中，分号表示矩阵中行的结尾。如用在每行的结尾，则 MATLAB 不会显示该行运算的结果，此功能可以用在 M 文件中控制命令的显示，减少输出篇幅。

4.4 关系运算与逻辑运算

在程序流控制中，常有一些逻辑和模糊逻辑的推理，需要对其作出“是真、是假”的回答。为此，MATLAB 提供了强大的关系运算和逻辑运算。虽然其他程序语言中也有类似的关系和逻辑运算，但 MATLAB 作为一种比较完善的科学计算环境，有其自身的特点。

在关系和逻辑运算中，MATLAB 有以下规定：

- 在所有的关系表达式和逻辑表达式中，输入的任何非 0 数都被看作是“逻辑真”，而只有 0 才被认为是“逻辑假”；
- 所有关系表达式和逻辑表达式的计算结果是一个由 0 和 1 组成的“逻辑数组（Logical Array）”，数组中的 1 表示“真”，0 表示“假”；
- 逻辑数组是一种特殊的数值数组，与“数值类”有关的操作和函数对它也适用；但它又不同于普通的“数值”，它还表示对事物的判断结论“真”与“假”，有其自身的特殊用途。

4.4.1 关系运算

MATLAB 关系操作符用来比较两个同样大小的数组，或用来比较一个数组和一个标量。其关系操作符如表 4-6 所示，其中包括所有常用的比较操作。

表 4-6 关系操作符

关系操作符	功能说明	关系操作符	功能说明
==	等于	<=	小于等于
~=	不等于	>	大于
<	小于	>=	大于等于

MATLAB 的关系操作符的运算法则：

- 当两个变量是标量 a 和 b 时
  - ◆ 若 a、b 之间关系成立，则关系运算结果为 1；
  - ◆ 若 a、b 之间关系不成立，则关系结果为 0；
  - ◆ 当两个维数相同的数组 A 和 B 比较时，数组 A、B 比较的是相同位置的元素，按标量的运算规则逐个进行。关系运算的结果是一个维数和 A 相同的数组，它的元素由 0 和 1 组成。
- 当一个数组 A 和一个标量 b 比较时

把标量 b 和数组 A 的每一个元素按标量关系运算规则逐个比较。关系运算的结果是一个维数和数组 A 相同的数组，它是由 0 和 1 组成。

- 优先级

由高到低为算术运算、关系运算和逻辑运算。

【例 4-4】关系运算示例。

```
A=[3 4 8;9 0 2;5 3 7]     %输入矩阵A
B=[4 4 1;7 8 4;5 1 7]     %输入矩阵B
```

```
A =
     3     4     8
     9     0     2
     5     3     7
B =
     4     4     1
     7     8     4
     5     1     7
E=(A==B)            %比较矩阵 A 和 B 是否相等
E =
     0     1     0
     0     0     0
     1     0     1
NE=(A~=B)           %比较矩阵 A 和 B 是否不等
NE =
     1     0     1
     1     1     1
     0     1     0
A0=(A>5)            %标出矩阵 A 中大于 5 的元素
A0 =
     0     0     1
     1     0     0
     0     0     1
B0=(B<=6)           %标出矩阵 B 中小于等于 6 的元素
B0 =
     1     1     1
     0     0     1
     1     1     0
```

4.4.2　逻辑运算

逻辑运算和逻辑函数在计算机语言中普遍存在。逻辑运算符提供了一种组合或否定关系表达式。在 MATLAB 中逻辑操作符如表 4-7 所示。

表 4-7　逻辑操作符

逻辑操作符	功能说明
&	与、和
|	或
~	否、非

下面是 MATLAB 中逻辑操作符的运算法则。

- 在逻辑运算中，非 0 元素的逻辑量为“真”，用 1 表示；0 元素的逻辑量为“假”，用 0 表示。
- 如果两个标量 a 和 b 运算，则：
    - ◆ a&b 当 a，b 全是非 0 时，运算结果是 1，否则是 0；
    - ◆ a|b 当 a、b 中只要有一个非 0，运算结果为 1；
    - ◆ ~a 当 a 是 0 时，运算结果是 1，否则是 0。
- 如果两个维数相同的数组 A 和 B 参与运算，则将数组 A 和 B 相同位置上的元素按标量的运算规则逐个进行运算。逻辑运算的结果是返回一个由 0 和 1 组成的与数组 A 具有同样维数的数组。
- 如果标量 b 和数组 A 参与运算，则：
    - ◆ 将标量 b 和数组 A 中的每个元素进行逻辑运算。逻辑运算的结果是返回一个由 0 和 1 组成的与数组 A 具有同样维数的数组；
    - ◆ 逻辑“非”运算是一元运算符，服从数组运算规则；
    - ◆ 在逻辑“与”、“或”、“非”三者中，“与”与“或”具有相同的优先级，从左向右依次执行，而都低于“非”的优先级。
- 通过增加“（）”可以改变各操作符之间的优先级。

下面通过例 4-5 来理解各个逻辑操作符的运算法则。

【例 4-5】逻辑操作符的用法。

```
A=[3 4 8;9 0 2;5 3 7]          %输入矩阵A
A =
     3     4     8
     9     0     2
     5     3     7
B=[4 4 1;7 8 4;5 1 7]          %输入矩阵B
B =
     4     4     1
     7     8     4
     5     1     7
AB=A&B                         %对矩阵A和B求逻辑“与”
AB =
     1     1     1
     1     0     1
     1     1     1
A_B=A|B                        %对矩阵A和B求逻辑“或”
A_B =
     1     1     1
     1     1     1
     1     1     1
C=~A                           %对矩阵B求逻辑“非”
```

```
C =
     0     0     0
     0     1     0
     0     0     0
cc=(A>3)&(A<6)            %求矩阵A中大于3小于6的元素
cc =
     0     1     0
     0     0     0
     1     0     0
```

4.4.3 关系与逻辑函数

除了上面的关系与逻辑操作符外，MATLAB 还提供了大量的其他关系与逻辑函数。这些函数在交互运算及进行矩阵的标化中非常有用，可以很方便地查找或替换矩阵中满足一定条件的部分或所有元素，如表 4-8 所示。

表 4-8 关系与逻辑函数

函数名称	功能简介
xor(*A*,*B*)	异或运算。*A* 和 *B* 对应元素同为 0 或非 0 时，相应位置元素取 0，否则取 1
any(*A*)	只要向量 *A* 中有非 0 元素，结果就是 1，否则结果是 0。矩阵 *A* 中的每一列有非 0 元素，结果为 1，否则为 0
all(*A*)	当向量 *A* 的所有元素全是非 0，返回 1；否则返回 0。当矩阵 *A* 中的每一列所有元素全是非 0，返回 1，否则返回 0
isequel(*A*,*B*)	当 *A*，*B* 对应元素相等时，相应元素位置取 1，否则取 0
ismember(*A*,*B*)	*A* 的元素是属于 *B* 集时，相应 *A* 元素位置取 1，否则取 0

下面将详细地阐述一些常用的函数的用法。

- any 函数

这是一个判断向量中元素是否有非 0 的函数。在矩阵处理时，有时要判断矩阵中的元素有无 0 值。如果在对矩阵进行数组除时，就要判断作除数的矩阵是否有 0 元素。其调用格式为：

- ◆ any(A)　若 *A* 是向量，如果 *A* 向量中至少有一个元素为非 0 数，any（*A*）将返回逻辑“真”，即为 1，否则为 0；若 *A* 为矩阵，函数 any(*A*)按向量的列判断，如果矩阵 *A* 的某列中存在某个元素为非 0 数，则返回当前列的结果为 1；若 *A* 是多维矩阵，any(*A*)将第一个不是单维的维作为向量，按向量的运算规则进行判断。
- ◆ any(*A*,dim)　指定的第 dim 维作为向量进行计算。如 any(*A*,1)就是按向量 *A* 第一维进行计算。

【例 4-6】函数 any 的用法。

```
A=[3 4 8;9 0 2;5 3 7]   %输入矩阵A
A =
     3     4     8
     9     0     2
```

```
     5     3     7
any(A)                  %逻辑运算
ans =
     1     1     1
any(A,2)                %对的二维进行逻辑运算
ans =
     1
     1
     1
```

- all 函数
  - all(*A*) 若 *A* 是向量，如果 *A* 向量中每个元素都是非 0 数，all(*A*)将返回逻辑“真”，即为 1，如果至少有一个元素为 0，则返回值为 0。若 *A* 为矩阵，函数 all(*A*)按向量的列判断，如果矩阵 *A* 的某列中所有元素都为非 0 数，则返回当前列的结果为 1。若 *A* 是多维矩阵，all(*A*)将第一个不是单维的维作为向量，按向量的运算规则进行判断；
  - all(*A*,dim) 指定的第 dim 维作为向量进行计算。如 all(*A*,1)就是按向量 *A* 第一维进行计算。

下面举例子来说明函数 all 的用法。

【例 4-7】判断矩阵 *A*=[3 4 8;9 0 2;5 3 7] 的所有元素是否都大于或等于 1。

```
A=[3 4 8;9 0 2;5 3 7]     %输入矩阵 A
A =
     3     4     8
     9     0     2
     5     3     7
all(all(A>=1))             %用函数 all 判断
ans =
     0
A>=1                       %看看矩阵 A 大于等于 1 的值
ans =
     1     1     1
     1     0     1
     1     1     1
all(A>=1)                  %用一次函数 all 判断
ans =
     1     0     1
```

从本例可以看出，函数的某些查找或判断功能非常强大，要仔细体会 all 函数的用法。

除了这些关系与逻辑函数外，MATLAB 还提供了大量的判断函数，以测试特殊值或条件的存在，返回逻辑值。这些函数如表 4-9 所示。

表 4-9　判断函数

函数名称	功能简介
find(x)	找出向量或矩阵 x 中非零元素的位置和标识
isfinite(x)	对应 x 中有限大小元素的位置取 1，其余取 0
isempty(x)	如 x 是“空”，结果为 1
isinf(x)	对应 x 中无穷大元素的位置上取 1，其余取 0
isletter(x)	对应 x 中英文字母的元素位置上取 1，其余取 0
isnan(x)	对应 x 中非数 NaN 元素的位置上取 1，其余取 0
isprime(x)	对应 x 中质数元素位置上取 1，其余取 0
isreal(x)	对应 x 中实数元素位置上取 1，其余取 0
ispace(x)	对应 x 中空格的元素位置上取 1，其余取 0
isa(x,'name')	name 是指具体数据类型的英文名称。假如 x 是所指类型，取值为 1
iscell(x)	若 x 是单位数组，则结果为 1
iscellstr(x)	若 x 是字符串组成的单位数组，则结果为 1
ischar(x)	若 x 是字符串，则结果为 1；否则为 0
isfield(x,'name')	若 name 是指定的名称并且是构架 x 的域名。则取值为 1
isglobal(x)	若 x 是全局变量，则结果为 1
ishandle(x)	若 x 是句柄代号，则结果为 1
islogical(x)	若 x 是逻辑数，则结果为 1
isnumeric(x)	若 x 是数值，则结果为 1
isobject(x)	若 x 是对象，则结果为 1
isspanse(x)	若 x 是稀疏阵，则结果为 1
isstruct(x)	若 x 是构架，则结果为 1
ishold	若当前图形保持状态是“ON”，则结果为 1
isseee	若计算机执行 IEEE 算法规则，则结果为 1
isstudent	若 MATLAB 是学生版，则结果为 1

这些命令在 MATLAB 程序设计和直接交互运算中非常有用。用户可以从指南或联机帮助中找到具体的调用格式。本书只将函数 find 作为例子进行较为详细地介绍，其余不再作具体解释。

函数 find 是找出向量或矩阵中非 0 元素的位置标识。在许多情况下，都需对矩阵或向量中某一特定条件下的元素位置进行定位，例如将某一矩阵中为 0 的元素位置取 1。如果这个矩阵的元素非常多，手工修改非常麻烦，而灵活运用 find 函数和各种逻辑关系运算可以实现绝大多数条件的元素定位。下面是其调用格式。

- I=find(X)　返回向量或矩阵 X 中的所有非 0 元素的位置标识组成的向量，如果没有非 0 元素则会返回空值。
- [I,J] = find(X)　返回矩阵 X 的非 0 元素行和列的标识。其中 I 是行标识，J 是列标识。此函数经常用在稀疏矩阵中。
- [I,J,V] = find(X)　返回矩阵 X 中的非 0 元素行和列的标识。其中 I 是行标识，J 是列标识，同时，将相应的非 0 元素的值放入列向量 V 中。即 I 和 J 的值与[I,J] = find(X)相同，只是增加了非 0 元素的值这一项。

下面举个例子来说明该函数的用法。

【例 4-8】函数 find 的用法。

A=[0 4 8;9 0 2;5 3 7]　%输入矩阵 A

```
A =
     0     4     8
     9     0     2
     5     3     7
find(A)                    %查找矩阵A中的非零元素位置
ans =                      %结果显示，非零元素的标识是按列进行的，即从第2列开始，
                             数完该列后再数第3列，依次数下去
     2
     3
     4
     6
     7
     8
     9
[I,J]=find(A)              %查找矩阵A中非零元素位置，并列出其具体位置
I =
     2
     3
     1
     3
     1
     2
     3

J =
     1
     1
     2
     2
     3
     3
     3
[I,J,V]=find(A)            %标出矩阵A中非零数值的位置，并写出非零数值
I=
     2
     3
     1
     3
     1
```

```
     2
     3
J =
     1
     1
     2
     2
     3
     3
     3
V =                                        %指出非零元素的数值
     9
     5
     4
     3
     8
     2
     7
find(A<3)                                  %找出矩阵 A 中小于 3 的元素的位置
ans =
     1
     5
     8
A(find(A==0))=-5                           %将矩阵 A 中等于 0 的元素替换为-5
A =
    -5     4     8
     9    -5     2
     5     3     7
C=[3 0 4; 0 7 5; 1 2 3]                    %输入矩阵 C
C =
     3     0     4
     0     7     5
     1     2     3
A(find(A==-5))=C(find(A==-5))              %将矩阵 A 中等于-5 的元素的值替换成矩阵 C
                                           中相应位置上的元素，即部分矩阵元素值的
                                           替换
A =
     3     4     8
     9     7     2
```

```
    5     3     7
A(find(A==9))=[]        %将矩阵 A 中等于 9 的元素删除
A =
    3     5     4     7     3     8     2     7
```

4.5 MATLAB 程序结构

计算机编程语言允许程序员根据某些结构来控制程序的执行次序。MATLAB 和大多数计算机语言一样，提供了设计程序所必须的程序结构，即顺序结构、循环结构和分支结构。在 MATLAB 中，循环结构由 while 和 for 语句实现，分支结构由 if 语句实现。

4.5.1 顺序结构

顺序结构就是依照顺序执行程序的各条语句。语句在程序文件中的位置反映了程序的执行顺序。MATLAB 的语句是由表达式构成。若程序是命令文件，程序运行完成后，中间变量都予以保留；若程序是函数文件，则程序运行完后，中间变量将被全部删除。

前面几章所举的例子大多是顺序结构，这里不再举例详述。

4.5.2 循环结构

循环是计算机解决问题的主要手段，在很多实际问题中会遇到许多有规律的重复运算和对某些语句的重复执行。在循环结构中，被重复执行的那一组语句就是循环体。它每循环一次，都需要作出是继续重复或是停止的判断，这个判断所依据的条件就是所谓的循环条件。MATLAB 语言提供了两种循环方式：for-end 循环和 while-end 循环。

1. for-end 循环

for 循环将循环体中的语句以固定和预定的次数进行重复执行。循环的次数一般情况下是已知的，除非用其他语句将循环提前结束。for 循环的语法格式为：

```
for x=array
   (commands)
end
```

for 指令后面的变量 x 称为循环变量，commands 为循环体。for 指令后面的数组 array 列数决定了循环体被重复执行的次数。也就是说，循环变量依次取数组的各列，对于每个变量值，循环体被执行一次。

数组 array 可以是整数、小数，还可以是负数，也可以是字符串、字符串矩阵或字符串组成的单元阵，不论它们取何值，都必须满足工程数组（向量）的条件，x 将穷尽此数组的取值，但作为取值的单元是有所不同的，如字符串是以每个字符作为取值单元；单元阵是以单元阵的元素为取值单元；数值矩阵是以列向量作为取值单元等。

【例 4-9】一个简单的 for 循环示例。

```
for I = 1:10
    A(I) = 1/(I+1)
```

```
    end
A =
    Columns 1 through 7
    0.5000    0.3333    0.2500    0.2000    0.1667    0.1429    0.1250
    Columns 8 through 10
    0.1111    0.1000    0.0909
```

说明：

- for 循环不会因为在循环体内对循环变量重新赋值而终止。
- 语句 1:10 是一个标准的 MATLAB 数组创建语句。在 for 后面的表达式中的数组可以是任何合法的 MATLAB 数组。
- for 循环结构可按需要嵌套使用。

为了得到高效代码，应尽量提高代码的向量化程度，避免使用循环结构。换而言之，如有一个等效的数组方法来解给定的问题时，应尽量避免使用 for 循环。如例 4-9，可以用下面的语句重写：

```
i=1:10;
A(i)=1./(i+1)
A =
    Columns 1 through 7
    0.5000    0.3333    0.2500    0.2000    0.1667    0.1429    0.1250
    Columns 8 through 10
0.1111    0.1000    0.0909
```

两种方法得出同样的结果，但后者执行的速度更快、更直观，输入较少。

技巧：为了得到最快的运算速度，在循环指令之前应尽量对数组进行预定义。

在例 4-9 中第一种情况下，for 循环体内每执行一次命令，变量 *i* 增加 1，这就使 MATLAB 每通过一次循环要花费时间，对变量 *i* 就要分配更多的内存空间。为此，例 4-9 中的程序可改为：

```
i=1:10;
A=zeros(1,10);
for i=1:10
        A(i)=1./(i+1)
end
```

而此时程序只需对 *A*(*i*)进行改变。

2. while-end 循环

while 循环是将循环体中的语句循环执行不定次数。其语法形式为：

```
while expression
      statements
end
```

在 while 循环结构中，statements 是循环体。当 MATLAB 碰到 while 指令时，首先检测表达式 expression 的值，如果其值为逻辑真（非 0），则执行循环体的内容，执行后再判

断 expressions 是否为真，若表达式的值仍为真，循环继续进行；一旦表达式的值为假，就结束循环。

【例 4-10】用 while 循环计算 1～100 之间整数的和。

```
sum=0;
i=1;
while i<=100
        sum=sum+i;
    i=i+1;
end
sum
sum =

        5050
```

说明：

- while 循环和 for 循环的区别在于，while 循环结构的循环体被执行的次数不确定，而 for 循环结构中循环体的执行次数确定；
- 一般情况下，表达式的值都是标量，但是，MATLAB 允许它是一个数组，此时只有当该数组所有元素均为真时，MATLAB 才会执行循环体；
- 如果 while 指令后的表达式为空数组，MATLAB 认为表达式值为假而不执行循环体。

4.5.3　分支结构

在计算中常常要根据不同的条件来执行不同的语句，当某些条件满足时，只执行其中的某一条或某几条命令，在这种情况下要用到分支结构语句。在 MATLAB 中提供了两种分支语句，即 if-else-end 和 switch-case-end 语句，它们各有特点。

1. if-else-end 分支结构

if-else-end 指令为程序流提供了一种分支结构，它有多种形式，其中最简单的调用形式为：

if expression

statements

end

if 后面的 expression 是表达式，当逻辑表达式的值为真时，则执行该结构中的执行语句内容，执行完后继续向下进行；若逻辑表达式的值为循时，跳过结特中的执行语句继续向下进行。

【例 4-11】折扣问题。

```
book=20;
number=40;
sums=0.0;
if number>=30
```

```
    sums=book*number*0.7;
end
sums
sums =
        560
```

当遇到有两种选择时，MATLAB 将采用下面的结构：

```
if expression
            statements1
    else
            statements2
end
```

在此结构中，statements1 和 statements2 是语句 1 和语句 2。此种调用格式的执行方式为：如果逻辑表达式的值为“真”时，则执行语句 1，跳过语句 2 向下执行；如果逻辑表达式测值为“假”时，则执行语句 2，然后向下继续执行。

当选择项多于两个时，MATLAB 将采用下面的结构：

```
if expression1
    statements1
elseif expression2
    statements2
        ……
        else
    statementsn
end
```

此种调用格式的执行方式为：如果逻辑表达式 1 的值为“真”，则执行语句 1，然后结束此结构；如果为“假”，则判断逻辑表达式 2 的值是否为真，如采为真，则执行语句 2，然后结束此结构；否则向下执行逻辑表达式 3……依此类推。

下面举一个例子来具体说明它们的用法。

【例 4-12】绘出函数 $y=\begin{cases}2x^2+1 & x\geq 1\\ 0 & -1<x<1\\ -x^3 & x\leq 1\end{cases}$ 的图像。

其程序为：

```
x=-3:0.1:3;
if x>=1
   y=2*x.^2+1;
   plot(x,y)
elseif -1<x<1
   y=5;
   plot(x,y)
```

```
else
      y=-x.^3;
      plot(x,y)
end
```

其执行结果如图 4-3 所示。

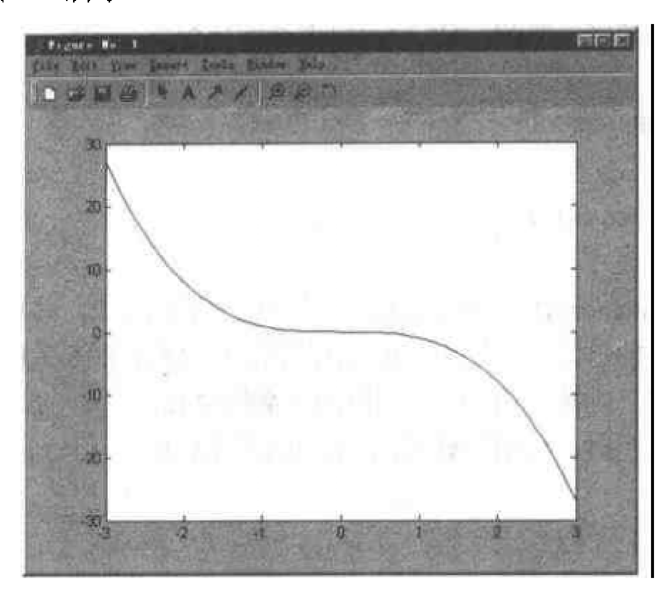


图 4-3 编程绘制函数图像

2. switch-case-end 分支结构

switch 语句是多分支选择语句，虽然在某些情况下其功能可以由 if 语句的多层嵌套来完成，但是会使程序变得复杂和难以修改，而用 switch 语句构造多分支选择结构时就显得更加简单明了、容易理解。switch 语句的一般调用格式为：

```
switch switch_expr
        case case_expr,
          statement, ..., statement
        case {case_expr1, case_expr2, case_expr3,...}
          statement, ..., statement
      ...
        otherwise,
          statement, ..., statement
 end
```

此种格式的执行方式为：

- switch 指令后面的表达式可以为任何类型，如字符串和标量等。对于标量，可按规则：表达式值= =检测值 i；而对于字符串，MATLAB 将调用函数 stremp 来完成比较：stremp(表达式，检测值 i)。有时表达式还可以是单元数组，此时 MATLAB 将把表达式的值和单元数组中的所有元素进行比较，如果单元数组中的某个元素和表达式的值相等，MATLAB 认为此次比较结果为真，从而执行与该检测值相

应的语句。

- 当遇到 switch 语句时，MATLAB 将表达式的值依次和各个 case 指令后面的检测值进行比较，如果比较结果为真，MATLAB 将执行 case 后面的语块，然后跳出该结构；如果比较结果为假，MATLAB 则取下一个检测值再比较。如果所有的结果都为假，即表达式的值与所有的检测值都不相等，MATLAB 将执行 otherwise 后面的语块。由此可见上述结构保证了至少有一组命令会得到执行。
- 每个 case 后面的检测值可以有多个，而且类型可以不同，每个 case 后面的检测值可以重复，这在语法上没有错误，只是在执行时，后面符合条件的 case 语句将被忽略，不起作用。
- 与 if 语句不同的是，各个 case 和 otherwise 语句出现的前后顺序并不会影响程序的执行结果。

【例 4-13】查找字符串类型。

其程序段为：

```
str=input('please input a string of Method   ')
        switch lower(str)
          case {'linear','bilinear'}
            disp('Method is linear')
          case 'cubic'
            disp('Method is cubic')
          case 'uearest'
            disp('Method is nearest')
          otherwise
            disp('Unknown method.')
        eud
```

运行结果：

```
please input a string of Method          %MATLAB 命令窗口显示
'linear'                                 %较入字符串
str =                                    %运行结果
linear
Method is linear
```

4.6 程序流控制语句

在 MATLAB 的程序设计中，有时需要提前终止循环、跳出子程序、显示错误或警告信息以及显示批处理文件的执行过程等，这就要用到程序流控制命令。MATLAB 中提供了以下几条流程控制及错误显示。

- return 指令

通常被调函数执行完后，MATLAB 会自动把控制转到主函数或命令窗口。如果在被调函数中插入 return 指令，即可强制 MATLAB 结束执行该函数并把控制转出。

● pause 指令

pause 指令使程序运行停止，等待用户按任意键继续。该指令在程序调试及需要看中间结果时特别有用。pause 指令有两种用法：

◆ pause 暂停执行程序，等待用户按任意键继续；

◆ pause(*n*) 在继续执行前，暂停 *n* 秒。

● break 指令

break 指令包含最内层 while、for、if 语句终止循环。通过使用 break 指令，可以不必等循环的自然结束，而是根据循环内部另设的某种条件是否满足来决定是否退出循环或是否结束 if 语句。在很多情况下，该指令是必须的。

● input 指令

input 指令是提示用户从键盘输入数值、字符串或表达式，并接受输入。其常用的调用格式为：

◆ *R*=input('Message') 将用户从键盘键入的内容 Message 赋值给变量 *R*。Message 可以是数字或表达式，也可以是字符串，此时字符串两端必须输入单引号，按 Enter 键后就可把输入内容赋值给变量 *R*；

◆ *R*=inpuT('Message','s') 将用户从键盘键入的内容 Message 作为字符串形式赋给变量 *R*，此时输入的任何内容（不论是数字还是字符）一律被当作字符串赋给变量 *R*。

提示：当输入字符串需要转行时，可用符号 "\"（"\n" 的令义是转行，也就是代表按 Enter 键），如果要输入 "\"，则要用 "\\"。

● keyboard 指令

keyboard 指令与 input 指令一样。在遇到 keyboard 指令时，MATLAB 将会暂停程序的运行，并调用键盘命令进行处理，用户可以从键盘输入各种 MATLAB 的合法命令。只要输入 return 指令，按 Enter 键后，程序将继续运行。

提示：keyboard 指令与 input 指令不同之处在于：它允许输入任意多个 MATLAB 指令；而 input 指令只能输入赋给变量的"值"，即数值、字符串或单元数组。

● error 和 warning 指令

在编写 M 文件时，常用到警示指令 error，其调用格式为：

error('message')

该格式的作用是显示错误信息并中止当前程序的运行，将控制返回键盘。

相关的命令还有：

◆ errortrap 错误发生后程序是否继续执行的双位开关；

◆ lasterr 显示 MATLAB 自动判断的最新出错原因，并中止程序；

◆ warning('message') 显示警告信息 message，程序继续运行；

◆ lastwarn 显示 MATLAB 自动给出的最新警告程序，并继续运行。

提示：error 指令与 break 及 return 指令是有区别的。break 指令是终止所在的最内层的循环；return 是终止其所在函数的运行，并将控制返回上一级函数或系统；而令令 error

是中止当前正在运行的程序并将控制返回到键盘上，不论它在哪一层函数中被调用，都会终止整个程序。如果错误信息是空字符串，则 error 指令不起作用。

- echo 命令

echo 指令通常用来控制 M 文件在执行过程中显示与否，这对程序的调试和演示极为有用。在使用过程中，对于命令文件和函数文件，echo 指令的作用方式不同。

对命令文件起作用的 echo 指令为：

- ◆ echo on 显示其后所有执行文件的指令；
- ◆ echo off 关闭其后所有执行文件的指令显示；
- ◆ echo 在上面两种状态之间转换。

对命令文件和函数文件都起作用的 echo 指令为：

- ◆ echo filename on 使 filename 指定文件的指令在执行时显示出来；
- ◆ echo filename off 关闭 filename 指定文件的指令显示；
- ◆ echo on all 使当前内存中函数文件的指令在执行时显示出来；
- ◆ echo off all 使其后所有函数文件的指令在执行时不再显示。

4.7 M 文 件

MATLAB 有两种常用的工作模式：一种是在工作空间窗口中直模输入简单的命令；另一种是 M 文件的编程工作方式。对于前一种工作模式，适用于命令行比较简单、输入比较方便，并且处理的问题相对较为特殊、没有一定的重复性和普遍性、差错处理比较简单的情况。这种模式主要体现了 MATLAB 作为一种“数学演算和图模工具”的特点，在以上的学习中，我们已经体会到这种模式的优越性和方便之处，这正是其情数学计算工具和编程语言无法比拟的。但是，仅有这方面的功能还不足以体现 MATLAB 的强大功能。在进行大量的重复性计算和输入时，单靠直接输入非常繁琐。而 MATLAB 接供的另一种工作模式则完美地解决了这方面的问题。本节着重介绍与 M 文件的编程工作模式有关的内容。

4.7.1 M 文件情介

MATLAB 是一个强有力的操作环境，它接中了 MATLAB 接供的完整而易用的输程语言。从形式上讲，MATLAB 程序文件是一个简单的 ASCII 码标准文本文件，扩展名一律用“.m”的形式。因为是文本文件，所以任何文字处理软件都可以对它进行编写和修改；从特征上讲，MATLAB 的语法比一般的高级语言情要简单，程序容易调试，人机交互性强。MATLAB 是解释性编程语言，即逐句解释运行程序。MATLAB 在初次运行 M 文件时将它的编程代码装入内存中，此过程会大大降低程序的运行速度，但这一缺点仅仅表现在初次运行时，再次运行该程序时便可直接从内存中取出代码运行，这又加情了程序的运行速度；从功能上讲，M 文件大大扩展了 MATLAB 的能力。MATLAB 的各种工具箱都由 M 文件组速。从这一点来说，若不了解 M 文件，MATLAB 的程大功能仅应用微小的

一部分。

MATLAB 提供了 M 文件编辑器作为编制和调试 M 文件的工作界面，在 MATLAB 的命令窗口中，用鼠标单击菜单栏上的【File】▸【Open】命令，选择【M-file】项，或者直接单击工具栏的【新建】按钮，进入 MATLAB 的 M 文件编辑器，如图 4-4 所示。

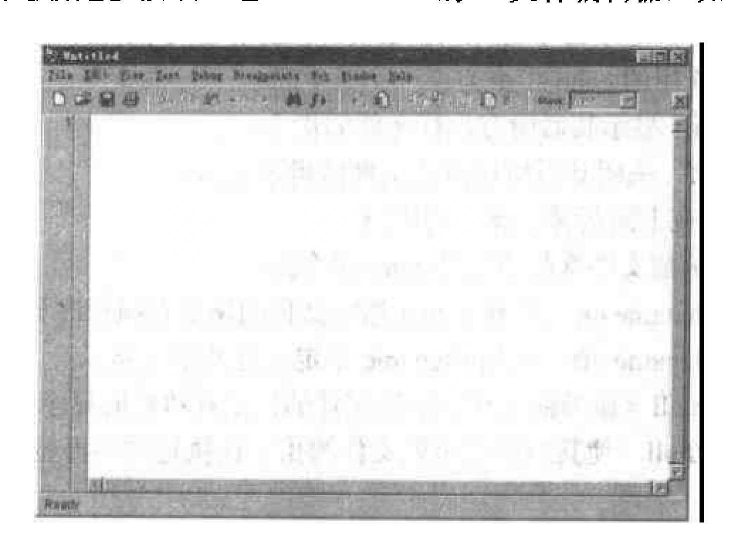


图 4-4 MATLAB 的 M 文件编辑器

M 文件编辑器同样也是一个 Windows 的标准界面，各个菜单项的使用同 MATLAB 的命令窗口大致相同。但比起 MATLAB 6.0 的工作环境窗口，M 文件编辑器的菜单栏多了【Text】、【Debug】和【Breakpoint】三项，分别用于对 M 文件的文字说明、调试和断点设置，关于菜单的详细用法，将在 4.8 节进行详细介绍。

除 M 文件编辑器外，还可以使用任何一种文字编辑器来创建 M 文件，只要将编写的文件在保存时加上“.m”后缀，同样可以使用 MATLAB 的 M 文件编辑器进行修改和调试。如果要在 MATLAB 的命令窗口中显示 M 文件的内容，使用函数 type 即可。

例如，要在 MATLAB 的命令窗口中显示 C4L12 文件的内容，则直接键入 type C4L12。

利用 M 文件可以自变函数和命令，也可以对已经存在的函数和命令进行修改和扩充，因此对 MATLAB 的二次开发非常方便。在 MATLAB 中，M 文件有两种形式，一种是命令文件（脚本文件 Script-file）；另一种是函数文件（Function-file）。下面将较为详细地介绍这两种 M 文件。

4.7.2 命令文件

如果要输入较多的命令，而且要经常对这些命令重复输入，利用命令文件将显得比较简单方便。建立命令文件的方法很简单，就是将要输入的所有指令按顺序放到一个扩展名为“.m”的文本文件中，而不需要预先定义。每次运行时只要输入 M 文件的文件名即可。实质上，运行一个命令文件等价于从命令窗口中按顺序连续运行文件中的指令。

命令文件中的语句可以访问 MATLAB 工作空间（Workspace）中的所有变量和资料，在命令文件运行过程中产生的所有变量都等价于直接从 MATLAB 工作空间建立而产生的所有变量，所以它们均是全局变量，任何其他的命令文件和函数都可以访问这些变量。这些变量一旦产生，就一直保存在内存中，除非用户用 clear 命令将它们清除。

注意：命令文件要放在 MATLAB 的搜索路径下，并且文件名最好不要与 MATLAB 的内置函数和工具箱中的函数名重名，以免产生混淆或发生执行错误命令等。

提示：命令文件中的符号“%”是引导的注释行，不予执行。

运行命令文件之前，必须将文件放在 MATLAB 的搜索路径上。最简单的方法是在 MATLAB 命令窗口中先将路径切换到存放文件的工作目录上，如 cd d:\mywork 等，或按照第1章所讲的方法将其加到 MATLAB 的搜索路径上。

下面我们举一个例子来说明命令文件的建立和运行。

【例 4-14】建立命令文件，并绘制宝石项链图。

（1）在 MATLAB 的命令窗口中，用鼠标单击菜单栏上的【File】按钮，单击【Open】命令，选择【M-file】项，或者直接单击工具栏的【新建】按钮，进入 MATLAB 的 M 文件编辑器。

（2）在编辑器窗口中输入文件内容：

```
t=(0:0.02:2)*pi;
x=sin(t);
y=cos(t);
z=cos(2*t);
plot3(x,y,z,'b-',x,y,z,'bd')
view([-80,60])
box on
legend('链子','宝石'):
```

（3）单击【File】▸【save】命令，MATLAB 系统将所写文件自动保存在磁盘目录 D:\MATLAB11\work 上，并取名为 C4L13.m。

（4）在 MATLAB 命令窗口中直接输入文件名 C4L13.m，运行结束后即可得到如图 4-5 所示的宝石项链图。

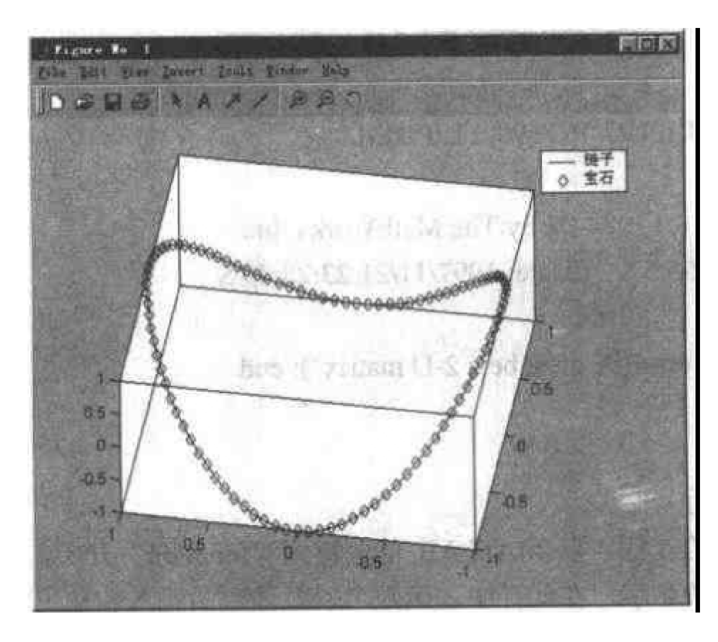


图 4-5 宝石项链图

4.7.3 函数文件

如果 M 文件的开头第一行是 funtion，则此文件是函数文件（Funtion File）。在 MATLAB 中提供的大部分命令都由函数文件定义，从使用角度上看，函数文件犹如一个“黑箱”，从外界只能看到传给它的输入变量和送出来的计算结果，而内部运作是藏而不见的。函数文件和命令文件的区别在于：命令文件的变量在文件执行完程序后仍然保留在内存中，而函数文件内定义的变量仅在函数文件内部起作用，当函数文件执行完程序后，这些内部变量将被清除。

函数文件和命令文件类似之处在于：它们都有一个扩展名为“.m”的文本文件，而且函数文件和命令文件一样，都是由文本箱辑器所创建的外部文本文件。

1. 函数文件的一般结构

MATLAB 的函数 M 文件通常由以下五部分组成：

- 函数定义行（Function Declaration Line）；
- H1 行（The First Help Text Line）；
- 在线函数帮助文件（Help Text）；
- 注释；
- 函数体（Function Body）。

我们结合一个例子来介绍函数文件的一般构成。下面是一个 MATLAB 提供的左右翻转的函数，其文件名为 rot90.m。整个函数内容为：

```
function y = fliplr(x)
%FLIPLR Flip matrix in left/right direction.
%    FLIPLR(X) returns X with row preserved and columns flipped
%    in the left/right direction.
%
%    X = 1 2 3      becomes   3 2 1
%        4 5 6                6 5 4
%
%    See also FLIPUD, ROT90, FLIPDIM.

%    Copyright (c) 1984-98 by The MathWorks, Inc.
%    $Revision: 5.5 $   $Date: 1997/11/21 23:28:49 $

if ndims(x)~=2, error('X must be a 2-D matrix.'); end
[m,n] = size(x);
y = x(:,n:-1:1);
```

- 函数定义行

位于函数文件的首行，以 MATLAB 的关键词“function”开头，把 M 文件定义为一个函数，并都明它的名字，在此行中，函数名以及函数的箱入和输出都数都被定义。

注意：函数 M 文件的函数名和保存文件名必须相同，如果两者不一致，MATLAB 将

忽略文件首行的函数定义名而以保存的文件名为准。函数的文件名必须以字母开头，后面可以是字母、下划线以及数字的任意组合，但不得超过31个字符。

函数 rot90 的定义行为：

function y = fliplr(*x*)

其中，“rot90”是函数名，输入参数为“*x*”，输出参数为“*y*”。

如果函数有多个输入、输出参数，则参数之间用逗号分隔，多个输出参数用方括号括起来。例如：

```
function [ans1,ans2,ans3]=axis(varargin);
function [xx,yy,zz] = meshgrid(x,y,z)
```

● H1 行

H1 行就是指定义行之后以“%”开头的第一注释行。按 MATLAB 自身文件的规定，H1 行包括大写体的函数名和运用关键词简要描述函数的功能。该行提供 look for 关键词和 help 在线帮助。

函数 rot90 的 H1 行为：

%FLIPLR Flip matrix in left/right direction

● 在线函数帮助文本

H1 行及其之后的连续以“%”开头的所有注释行构成整个在线帮助文本。它通常包括行数输入、输出参数的含义以及调用格式的说明。

例如，函数 rot90 的帮助文本为：

```
%    FLIPLR(X) returns X with row preserved and columns flipped
%    in the left/right direction.
%
%    X = 1 2 3      becomes   3 2 1
%        4 5 6                6 5 4
%
%    See also FLIPUD, ROT90, FLIPDIM.


%    Copyright (c) 1984-98 by The MathWorks, Inc.
%    $Revision: 5.5 $   $Date: 1997/11/21 23:28:49 $
```

当输入“help 函数名”时，MATLAB 就会显示 H1 行和帮助文本。

● 注释

除了函数开始独立的帮助文本外，还可以在函数体中添加对语句的注释。注释必须以“%”开头，MATLAB 在编译执行 M 文件时把每一行中“%”后面的内容全部作为注释而不进行输译。

● 函数体

为清楚起见，它与前面的注释以空行相隔。这部分内容由实现 M 函数文件功能的 MATLAB 指令组成。函数体可以有流程控制、输入输出、计算、赋值和注释，还可以包括函数调用和对命令文件的调用。

例如：函数 rot90 的函数体为：

```
if ndims(x)~=2, error('X must be a 2-D matrix.'); end
[m,n] = size(x);
y = x(:,n:-1:1);
```

提示：在组成 M 函数文件的几部分中，定义行是 MATLAB 所必须的，其他各部分的内容可以没有，此时，这种函数就成为空函数。

2. 函数参数的传递

函数调用的过程就是参数传递的过程。如函数 rot90 中，有：

```
v=[3 4 4;4 6 2]
y=tor90(v)
```

这个调用过程就是把赋了值的变量“v”传递给函数中的输入参数“x”，然后把函数运算的返回值传递给输出参数“y”。

函数有它自己的专用工作空间，它与 MATLAB 的工作空间分开。函数内变量与 MATLAB 工作空间之间惟一的联系就是函数的输入、输出参数。函数内任意参数的改变并不影响 MATLAB 工作空间的变量。MATLAB 工作空间所定义的变量不会延伸到函数工作空间上；在函数体内定义的变量也不会延伸到 MATLAB 的工作空间内。

在函数 M 文件内可以调用命令文件。此种情况下，命令文件查看函数工作空间时，不会查看 MATLAB 的工作空间。从函数 M 文件调用的命令文件不必用调用函数编译到内存，函数每调用一次，它们就被打开和解释。因此，从函数 M 文件内调用命令文件减慢了函数的执行速度。

函数还可以递归调用，即函数能调用它们本身。递归调用函数功能在许多应用场合中非常有用。在编制递归调用函数时，必须确保会终止，否则 MATLAB 将陷入死循环。

下面举例来说明函数递归调用的使用。

【例 4-15】用递归调用的形式计算 n 的阶乘。

（1）在 M 文件编辑器中编写递归调用的函数文件 factor.m。

```
function y=ff(n)            %定义函数文件
if n==1
    y=1;
    return;
else
    y=n*ff(n-1);
    return;
end
```

（2）在 MATLAB 的工作窗口中运行函数文件 ff(n)。

```
ff(10)
ans =
     3628800
```

4.8 M 文件调试的主要功能

从程序设计的角度来讲，MATLAB 语言比其他程序设计语言在说明结构上要简单得多。但是，MATLAB 也有自己严密的语法，用户必须按语法的要求来编写 MATLAB 的程序，否则会产生一些错误。

在编写 MATLAB 程序中，错误一般有语法（Syntax）错误和运行（Run-time）错误。语法错误是指变量名、函数名的误写及标点符号的缺、漏等。对于这类错误，MATLAB 通常能在 P 码编译和运行时立即发现，终止其运行，并给出相应的错误原因以及所在的行号。运行错误由算法本身引起，发生在运行过程中，相对语法错误而言，动态的运行错误比较难处理。主要原因是算法模型与期望目标不一致、程序模型与算法不一致等。

对于编写程序时出现的错误，MATLAB 提供了 M 文件的调试（Debug）功能，可以对 M 文件进行调试。MATLAB 中有两种调试法：直接调试法和图形调试法。前一种调试法一般用于比较简单的程序。后一种调试法通常用在函数文件规模很大，文件内嵌套复杂，有较多函数、子函数以及私有函数调用，直接调试法可能失效的情况下。

4.8.1 调试的主要命令

在直接调试法中，MATLAB 提供了几种调试命令，几种主要的命令列于表 4-10 中。

表 4-10　　MATLAB 的调试基本指令

指令名称	指令功能
dbstop	设置程序点
dbclear/dbclear all	清除程序（所有）断点
dbcont	恢复程序运行至程序结束或到另一断点
dbdown/dbstep in	深入下层局部工作区
dbstack	列函数调用关系
dbstatus	列出所有断点
dbstep	制定执行步数
dbtype	列出行号内的 M 文件
dbup	向上层改变工作区
dbquit	退出调试状态

4.8.2 调试的使用

如发现函数 M 文件有错误，则可在函数中设置断点（Breakpoint），帮助分析和发现程序的错误原因。在调试时，若函数程序遇到断点，就将子程序暂时停下来，并将断点处的命令行显示出来，同时显示键盘输入的提示符。用户可以在该提示符之后输入任何合法的 MATLAB 命令。

注意：调试命令只能对函数 M 文件使用，不能对其他 M 文件使用。M 文件的端点与编译过的 M 文件相关联，如果 M 文件被清除或被重新编辑，则所有的断点即被取消。

4.8.3 利用编辑器修改和调试 M 文件

从 MATLAB 5.x 版本起，MATLAB 提供了更为简便的图形调试器（Graphical Debugger）。它与 M 文件编辑器集合成为一体，本节将着重讲述利用编辑器来对 M 文件进行调试。

需要注意的一点是，MATLAB 6.0 的 M 文件编辑器和 5.3 版本的编辑器略有不同，它新增加了【Text】、【Debug】和【Breakpoints】三个菜单项，其中所包含的子菜单项也略有不同（见图 4-4），同时工具栏上新增加了关于设置和清除断点、调试 M 文件共 7 个按钮。本节主要针对 MATLAB 6.0 的调试方法予以介绍。

【Text】菜单项的内容和命令的含义如表 4-11 所示。

表 4-11 【Text】菜单的命令和含义

菜单命令	快捷键	命令含义
Comment	无	将光标所在行设为说明语句
Uncomment	无	取消设为说明语句
Dcrease Indent	Ctrl+[	减小缩进量
Inrease Indent	Ctrl+]	增加缩进量
Balance Delimiter	Ctrl+B	选中最外边界内的内容
Smart Indent	Ctrl+I	取消缩进量
Evaluate Selection	F9	激活所选中的内容

【Debug】菜单项的内容和命令的含义如表 4-12 所示。除了【Go until Cursor】子菜单项外，其余 5 个命令在工具栏上均有相应的按钮与之对应。

表 4-12 【Debug】菜单的内容

菜单命令	快捷键	命令含义
Step	F10	单步执行当前行
Step In	F11	深入（被调）函数
Step Out	Shift+F11	跳出（被调）函数
Save & Run	F5	保存并运行
Go until Cursor	无	随光标前进
Exit Debug Mode	无	退出调试

注意：只有当文件处于调试状态时，调试菜单项的子菜单项才处于可选状态，否则这些命令呈灰色不可选状态。

【Breakpoint】菜单项的内容和命令的含义如表 4-13 所示。其中【Set/Clear Breakpoint】和【Clear All Breakpoint】两个子菜单项在工具栏上有相应的快捷按钮与之对应，可通过直接单击来执行相应命令。

表 4-13 【Breakpoint】菜单的内容

菜单命令	快捷键	命令含义
Set/Clear Breakpoint	F12	设置/清除断点
Clear All Breakpoint	无	清除所有断点
Stop If Error	无	如有错调停止程序运行
Stop If Warning	无	如有警告信息停止程序运行
Stop If NaN or Inf	无	如出现非数 NaN 或者无穷大 Inf，停止程序运行

4.9 小　结

MATLAB 作为一种高级应用软件，不仅具有命令行操作的直接交互方式，还有着自己的编程语言，就像 FORTRAN 语言、C 语言等编程语言一样。从本质上讲，MATLAB 6.0 也就是 MATLAB R12 版本的编程语言环境。通过本章的学习，应该能够掌握如下内容：

（1）了解 MATLAB 的变量类型，掌握基本表达式。

（2）了解 MATLAB 的数据结构，掌握 MATLAB 的字符串数组、单元数组以及结构数组的建立和访问方法。

（3）掌握 MATLAB 运算符与操作符的含义和用法。

（4）了解并初步掌握 MATLAB 的关系运算与逻辑运算。

（5）熟练掌握 MATLAB 的三种程序结构——顺序结构、循环结构和分支结构。

（6）掌握并善于利用 MATLAB 的控制流语句命令。

（7）学会 MATLAB 的 M 文件的编写方法，包括命令文件和函数文件。掌握 M 文件的调试命令和调试方法。

习　题

1．创建一个表达式 $z=\dfrac{\sqrt{4x^2+1}+0.5457e^{-0.75x^2-3.75y^2-1.5x}}{2\sin 3y-1}$，并求当 $x=1,y=2$ 时的 z 值。

2．创建一个包含“he is the best student with glasses in our class”字符串数组，并查找字符串“ass”的位置。

3．输入如下两个矩阵 A 和 B，对矩阵 A 和 B 作关系运算，标识出两矩阵中元素相等的位置，元素值不等的位置，并标识出矩阵 A 中所有小于 0 的元素。

$$A=\begin{bmatrix}1 & 2 & 3\\ -2 & 1 & 3\\ -3 & 2 & 1\end{bmatrix} \qquad B=\begin{bmatrix}1 & 4 & 3\\ 3 & 2 & 8\\ 5 & 2 & 3\end{bmatrix}$$

4．对第 3 题中的矩阵 A 和 B 作逻辑“或”、“与”运算，并标识出矩阵 B 中所有大于 2 并小于 5 的元素位置。

5．利用 for 循环求 1!+2!+3!⋯+20!的值。

6．用 while 循环求 1～200 之间的整数之和。

7．编写一个 M 文件，画出下列分段函数所表示的曲面。

$$p(x,y)=\begin{cases}0.54e^{-0.75x^2-3.75y^2-1.5y} & x+y>1\\ 0.7575e^{-x^2-6y^2} & -1<x+y\le 1\\ 0.5457e^{-0.75x^2-3.75y^2+1.5y} & x+y\le -1\end{cases}$$

8．编写一个求圆的面积的函数文件。

9．编写一个求圆的面积的命令文件。

第 5 章　MATLAB 符号计算及工具箱

在工程领域和科学研究中,符号运算占有很大比例，本章将介绍符号运算的概念、基本用法和 MAPLE 资源的调用，同时将详细介绍 MATLAB 的符号函数计算器。

除数值计算外，像公式推导、因式分解等这一类含有 x、y、z、α、β、δ 等符号变量的符号表达式的抽象运算，以及求解代数方程或微分方程的精确解等，在工程领域和科学研究中也占有很大比例。MathWorks 公司于 1993 年购买了主要针对符号计算、具有强大符号运算能力的 MAPLE V 软件的使用权，随后以 MAPLE 的内核为符号计算的“引擎”，依靠 MAPLE 已有的库函数（library）开发了在 MATLAB 环境下的实现符号计算的工具箱，即符号数学工具箱（Symbolic Math Toolbox），成功地将 MAPLE 的符号运算合成到 MATLAB 的数值计算环境中去。在 MATLAB 5.0 以上版本中开始使用面向对象编程的概念，MATLAB 5.3 以后的版本以 MAPLE V Release 5 为基础进行开发，同时采用了重载技术和一种全新的数据结构——符号对象（Symbolic Object），又称为 sym 对象，使符号运算和数值运算的结合日臻完美。

MATLAB 的符号数学工具箱包括基本符号数学工具箱和扩展符号数学工具箱两个子工具箱。其中基本符号数学工具箱是 MATLAB 语言的自然扩展，它集中了大约 100 多个 MATLAB 函数，这些是一些基本的函数命令，为调用 MAPLE 的“内核”提供了相应的命令，同时可以在这个工具箱内调用 MAIPLE 的线性代数工具包。而在扩展符号数学工具箱内可以调用所有非图形类的属于 MAPLE 的工具包，并且运用 MAPLE 的编程特征完成自设的运算。MATLAB 有了这两个子工具箱，用户便可以使用符号对象编写自己的 M 文件和函数，继续扩展 MATLAB 的强大功能。

符号数学工具箱一共有三个通道与 MAPLE 交换信息：

- 通过基本符号数学工具箱。即在用 MATLAB 语言编写的多个函数中，通过若干个专用函数进行符号运算。用于符号运算的专用函数按照内容可分为：
  - ◆ 函数体（Function Body）符号表达式和符号矩阵的操作；
  - ◆ 线性代数；
  - ◆ 微积分；
  - ◆ 符号方程的求解；
  - ◆ 多项式的化简、展开和代入；
  - ◆ 特殊的数学函数。
- 通过 maple.m、mpa.m 两个专门设计的 M 文件进行符号运算。这种符号运算的运算方式要求掌握一些 MAPLE 的基本语句；
- 通过 MATLAB 中的函数计算器（Function Caculator）也可以进行较为简单的符号运算，这是 MATLAB 最方便、最直观的符号运算方法。

在数值计算参与输入、输出和中间计算的过程中，所有运作的变量都是被赋了值的数

值变量；而在符号计算的过程中，参与运作的变量都是符号变量（Symbolic Variable）（包括符号表达式中出现的数字也当作符号处理）。使用字符串进行符号分析而不是基于数组的数值分析，是符号数学工具箱区别于其他工具箱的重要特征。

同样，可以在符号数学工具箱中使用【Help】命令或者使用网络浏览器查阅 HTML 帮助文件获得关于符号运算命令和函数的帮助信息。需要说明的是，由于【Help】命令查找的范围只限于 MATLAB 的内部函数和命令，因此限制了它的使用范围。当查找的命令不是 MATLAB 的函数，而是仅由它使用的属于 MAPLE V 的函数时，【Help】命令将无法得出结果。此时便要使用 MAPLE V 的帮助命令【Mhelp】进行查找，其使用格式与【Help】命令完全相同，但得到的结果是 MAPLE V 中函数命令的帮助信息。

5.1 创建符号变量

MATLAB 5.0 以上版本中的符号工具箱沿用数值计算的赋值模式规定：在进行符号计算时，首先要定义基本的符号对象（可以是常数、变量以及表达式等），然后利用这些基本符号对象去构成新的表达式，从而进行所需的符号运算。在运算中，凡是由包含符号对象的表达式所生成的新对象也都是符号对象。可以使用 sym 和 syms 这两个函数命令来创建和定义基本的符号对象。

5.1.1 sym 函数定义符号变量

sym 函数的调用格式如下：

- *S*=sym(arg)　从表达式 arg 创建一个 sym 对象 *S*，如果 arg 是一个字符串（string），则 *S* 是符号变量或符号数；如 arg 是数值标量或矩阵，则 *S* 是这些给定数值的符号形式。以下是 sym 函数调用形式的具体实现方式：
  - ◆ *x*=sym('*x*')　建立符号变量 *x*，变量的值为单引号内的字符或字符串，这里是和变量名相同的字符'*x*'；
  - ◆ *x* = sym('*x*','real')　设定符号变量为实型变量（Real），此时 conj(*x*)和 *x* 相等；
  - ◆ *x* = sym('*x*','unreal')　使 *x* 为纯粹的形式变量，没有附加属性。一般用来清除 *x* 的实型属性；
  - ◆ 类似 pi = sym('pi')和 delta = sym('1/10')　建立符号数，这种方式避免了浮点数本身的近似，建立的符号数是数值的精确表示。这种方式建立的符号数 pi 可以临时代替内置的同名数值函数 pi。
- *S* = sym(*A*,flag)　可以将数值或矩阵转化为符号形式，其中 flag 选项有四项参数，即'*f*','*r*','*e*' 和 '*d*.'，它们对应于不同的符号形式，'*r*'为缺省项。各项的含义如下：
  - ◆ 选项'*f*'　代表十六进制浮点形式。格式为：'1.*F* '*2^(*e*) or '-1.*F* '*2^(*e*)，其中 *F* 是由 13 位十六进制数组成的字符串，*e* 是整数（但 *F* 前面的“1”是十进制数）。例如：

```
sym(1/10,'f')
ans =
```

'1.999999999999a'*2^(-4)

因为 1/10 不能用浮点精确地表示。

◆ 选项'r' 代表有理数形式。像 p/q、$p*pi/q$、sqrt(p)、2^q 和 10^q 之类的有理数形式，有效地补偿了舍入误差，但是也有可能表示的浮点值和原值不相等。如果找不到简单的有理数形式近似，则可采用形式 p*2^q 产生正确的浮点数，其中 p 是很大的整数。例如：

```
sym(4/3,'r')
ans=
'4/3',
```

而 sym(1+sqrt(5),'r')

```
 ans=
    7286977268806824*2^(-51)
```

◆ 选项'e' 估计误差。根据 eps（浮点运算的相对精度）给出理论表达式和实际计算的误差。例如：

```
sym(3*pi/4,'e')
ans =
    3*pi/4-103*eps/249
```

◆ 选项'd ' 表示十进制小数。其有效数字位数由 digits 函数定义，缺省的有效位数是 32 位。如果有效位数小于 16 位，则会损失一些精度。例如：

```
digits(10), sym(4/3,'d')
ans =
1.333333333
digits(20), sym(4/3,'d')
ans =
1.3333333333333332593
```

可以看出，当有效位数超过 16 位时，结果不再以 3 的循环结尾，而是以最接近 4/3 的浮点数的精确十进制形式结尾。

5.1.2 syms 函数定义符号变量

在符号变量和变量值相同时，可以用 sym 的简捷方式来建立符号变量，即用函数 syms 来表达。该函数的用法及其与 sym 函数的关系如下：

- syms arg1 arg2 ...等价于 arg1 = sym('arg1'); arg2 = sym('arg2'); ...
- syms arg1 arg2 ... real 等价于 arg1 = sym('arg1','real'); arg2 = sym('arg2','real'); ...
- syms arg1 arg2...positive 等价于 arg1=sym('arg1','positive');
  arg2=sym('arg2','positive'); ...
- syms arg1 arg2 ... unreal 等价于 arg1=sym('arg1','unreal');
  arg2= sym('arg2','unreal'); ...

【例 5-1】创建一个字符型数据变量和一个符号型数据变量，并比较它们的不同。

```
f=sym('a')          %创建一个符号变量f
```

```
f =
a
f1='a'              %创建一个字符变量f1
f1 =
a
size(f)             %求符号变量f的大小
ans =
     1     1        %结果为一个1×1的矩阵
size(f1)            %求符号变量f的大小
ans =
     1     1        %结果同样为一个1×1的矩阵
f==f1               %对变量f和f1进行逻辑运算，看两变量是否相等
ans =
     1              %返回结果为1，表示"真"，二者内容是相等的
abs(f1)             %求字符型变量f1的ASIIC码值
ans =
    97
abs(f)              %求符号型变量f的ASIIC码值
ans =
abs(a)              %可见f和f1的ASCII码值是不相同的，因此两个变量是不同的
```

由本例题可以清楚地看出字符型变量和符号型变量的区别。鉴于符号型数据是符号运算的主要数据类型，本章将主要采用符号型数据作为介绍命令的参数。

5.2 创建符号

5.2.1 符号表达式和符号方程

符号表达式（Symbolic Expression）和符号方程（Suymbolic Equation）是将表达式和方程赋给一个符号变量，通过引用该符号变量来引用相应的表达式或方程。它们是两个不同的符号对象。符号表达式是代表数字、函数、算子和变量的字符串和字符串数组，不要求变量有预先确定的值，而符号方程是含有等号的表达式。它们的区别在于表达式不含等号，而方程必须带等号。创建符号表达式和符号方程有两种方法。

1. 采用 sym 命令

采用 sym 命令来创建符号表达式和符号方程的调用格式为：

f=sym('arg')

arg 代表一个表达式或方程，注意，不要遗忘单引号。

例如，创建一个表达式命令为：

```
f=sym('a*x^2+b*x+c')
```

```
f =
a*x^2+b*x+c
```

又如创建一个方程，其命令为：

```
f=sym('a*x^2+b*x+c=0')
f=
a*x^2+b*x+c=0
```

2. 直接法

符号表达式和符号方程也可以直接采用与 MATLAB 中字符串变量的创建方法一样来建立。如：

```
f='a*x^2+b*x+c'
f=
  a*x^2+b*x+c
```

创建一个方程，命令为：

```
f='a*x^2+b*x+c=0'
f=
  a*x^2+b*x+c=0
```

注意：符号表达式和符号方程对空格很敏感。因此，在创建符号表达式或符号方程时，不要在字符间任意加空格符；在符号计算中出现的数字也是当作符号处理的；符号矩阵是数组，其元素是符号表达式。

5.2.2 创建符号矩阵

符号矩阵的创建有两种方法，即由 sym 命令创建和由字符串直接输入创建。

1. 由 sym 命令创建符号矩阵

矩阵元素是不带等号的符号表达式，各矩阵元素的长度可以不同，矩阵行之间用分号隔开，各元素间用逗号或空格隔开。例如：

```
A=sym('[4+x x^2 x;x^3 5*x-3 x*a]')
A =
    [   4+x,    x^2,       x]
    [   x^3, 5*x-3,      x*a]
```

或利用简捷方式，命令如下：

```
syms x a
A=[4+x x^2 x ;x^3 5*x-3 x*a]
A =
    [   4+x,    x^2,       x]
    [   x^3, 5*x-3,      x*a]
```

2. 由字符串直接输入创建矩阵

此种输入法与 MATLAB 字符串矩阵的输入相似。它不需要调用 sym 函数，但要保证在同一列中各元素字符串有同样的长度，在较短的字符串前后用空格符填充。上面的矩阵也可以用下列命令来实现：

```
A=['[4+x x^2    x   ]';'[x^3 5*x-3 x*a]']
A =
    [4+x x^2    x   ]
    [x^3 5*x-3 x*a]
```

注意：此种方法在建立时要求符号矩阵每一行的两端都有方括号，而 MATLAB 字符串矩阵仅在首尾有方括号。

5.2.3 数字矩阵和符号矩阵的转换

由于 MATLAB 的数值型和符号型是两种不同的数据类型，因此在 MATLAB 中，这两个数据类型的变量之间不能直接进行符号运算，必须在 MATLAB 的工作空间内将数值型转换为符号型后才能进行符号运算。不管数值矩阵的元素是以分数或是浮点数表示，转换后的符号矩阵都将以最接近有理式的形式给出。数字矩阵转换成符号矩阵同样通过 sym 命令来实现。

【例 5-2】数值矩阵转化为符号矩阵。

```
A=[2/5 4/0.78 sqrt(23)/3;0.33 0.3333 log(4)]     %输入数值矩阵 A
A =
    0.4000    5.1282    1.5986
    0.3300    0.3333    1.3863
FA=sym(A)                                         %将数值矩阵 A 转化为符号矩阵 FA
FA =
    [ 2/5,          200/39,                 sqrt(23/9)]
    [ 33/100,       3333/10000,    6243314768165359*2^(-52)]
```

5.2.4 符号矩阵的引用和修改

在数值计算中，可以用一个指令来实现对矩阵中的任何一个子矩阵进行引用和修改，但在符号计算中，引用（Quote）和修改（Modify）只能对符号矩阵的元素一个一个地进行，其具体的用法如例 5-3。

了解了函数 sym 和 syms 的用法后，我们用例 5-3 来说明如何用这两个函数建立符号变量、表达式和矩阵。

【例 5-3】用函数 sym 和 syms 建立符号变量、表达式和矩阵。

（1）用命令建立符号变量 x 和 beta，并设置附加属性为实型变量，命令为：

```
x = sym('x','real');
beta = sym('beta','real');
```

如果采用简捷方式：

```
syms x beta real
```

结果为：

```
x =
    x
beta =
```

```
        beta
```

（2）建立复数变量 z：

```
z=x+beta*i;
```

用下面的命令计算相应变量的共轭复数：

```
conj(x)
conj(z)
expand(z*conj(z))
```

相应结果为：

```
ans =
        x
ans =
        x-i*beta
ans =
        x^2+beta^2
```

（3）建立三次函数 $y = ax^3 + bx^2 + cx + d$ 的符号表达式，其命令为：

```
y=sym('a*x^3+b*x^2+c*x+d')
```

结果为：

```
y =
    a*x^3+b*x^2+c*x+d
```

这个命令将符号表达式 $ax^3 + bx^2 + cx + d$ 赋值给变量 y，由于没有建立对应于表达式中 a、b、c、d 和 x 的变量，y 中的内容只是一个简单的字符串。为了使 y 成为一个真正的符号表达式，可以执行符号数学运算（微积分等），必须显式地建立这些变量。其命令如下：

```
a=sym('a')
b=sym('b')
c=sym('c')
d=sym('d')
x=sym('x')
```

或用简捷方式，命令如下：

```
syms a b c d x
```

如果用上述两个命令建立符号矩阵，命令为：

```
syms a b c d
A=[a a+c d+b;c d a+c;a+c+d c c+d*a]
```

结果为：

```
A =
      [        a,     a+c,     d+b]
      [        c,       d,     a+c]
```

```
[ a+c+d,        c, c+d*a]
```

如果想要用“eee”代替矩阵 A 中的 A（1，3）位置的元素，用“ddd”代替矩阵 A 中的 A（3，2）位置上的元素，命令为：

```
syms eee ddd
A(1,3)=eee;
A(3,2)=ddd;
A
```

结果为：

```
A =
    [        a,     a+c,     eee]
    [        c,       d,     a+c]
    [ a+c+d,       ddd, c+d*a]
```

5.2.5 建立符号数学函数

MATLAB 的符号数学工具箱具有创建数学函数的功能，既可以建立一般的数学函数，如 $y = ax^2 + bx + c$，又可以建立抽象的数学函数，如 $y = f(x)$ 等。

1. 建立一般的数学函数

MATLAB 的符号数学工具箱有两种方式建立一般的数学函数，即利用符号表达式和利用建立 M 文件的方法。

- 利用符号表达式

先定义符号变量，再建立符号表达式，例如：

```
syms x y z
f=sin(x+y)/(x-y)
g=sqrt(x^2+y^2+z^2)
```

可以使用符号数学工具箱的函数对函数 f、g 进行操作。

- 建立 M 文件

M 文件可以更方便地使用函数。MATLAB 的许多内置式和函数便是用 M 文件建立的。例如，要建立一个名为 sinx 的函数用于求 sin(x)/x 的值，只要建立一个求 sin(x)/x 值的 M 文件，然后放入设置的当前工作目录中即可（本文以 MATLAB 的缺省设置 D:\matlabR12\work 为当前工作路径）。

```
function y=sinx(x)
%SINX the symbolic sinc function to comput sin(x)/x
if isequal(x,sym(0))
    y=1;
else
    y=sin(x)/x;
end
```

同样，可以将此函数扩展到多变量的情况，用户可以用此种方法方便地建立自己的函

数库。

2. 建立抽象的数学函数

在实际工程应用和科学计算中，经常用 $f(x)$、$g(x, y, z)$或 f、g 之类的符号来代表一个已知或未知的抽象函数进行运算，此类运算同样可用 MATLAB 的 sym 命令完成。

用 sym 命令创建一个变量为 var1、var2、var3、…、varn 的 f 函数，格式为：

```
syms var1 var2 var3 … varn
f=sym('f(var1 var2 var3 … varn)')
```

可以采用工具箱中的命令对抽象函数 f 进行操作。

【例 5-4】 建立一阶差分函数 $f = \dfrac{f(dx+h)-f(x)}{h}$。

```
syms x h                  %建立符号变量
f=sym('f(x)')             %建立抽象函数f
f =
   f(x)
df=(subs(f,x,x+h)-f)/h    %建立一阶差分公式
df =
     (f(x+h)-f(x))/h
```

在进行符号积分变换时，使用 sym 的此项功能使运算变得十分方便。

5.2.6 三种数据类型之间的相互转换

MATLAB 中的数值型、字符型和符号型三种数据类型的等级不同，其中数值变量级别最低，字符变量级别居中，符号变量级别最高。如果有这三种变量参与的混合运算，系统将会把所有参与运算的变量自动统一转换为变量等级最高的类型，然后进行计算，也可以通过命令来完成对不同数据类型之间的转换。大致可以分为三种情况：

- 转换为数值变量
  - ◆ x=double(S) S 为符号变量时，将 S 转化为数值变量 x，若 S 中有非数字的符号，则系统给出出错提示；当 S 为字符变量时，将 S 转化为数值矩阵，矩阵中的元素为 S 中相应字符的 ASCII 码值；
  - ◆ x=str2num(S) 专门用于将字符型变量转换为数值变量。当 S 中含有非字符型变量时，该命令返回一个空矩阵；
  - ◆ x=numeric(S) 将 S 转换为数值型，不管 S 是字符变量还是符号变量。但 S 不能是矩阵，否则给出错误信息。
- 转化为符号变量

命令为 S=sym(f)，此命令对变量 f 没有任何限制，只要不是非法的表达式或字符矩阵即可。

- 转化为字符变量
  - ◆ S=int2str(x) 将整数 x 转化为字符变量 S。当 x 为普通有理数时，将对 x 四舍五入之后进行转化；当 x 为虚数时，sym(f)仅对实部进行转换；
  - ◆ S=num2str(x) 将普通数值变量转化为字符变量 S。此命令对 x 的限制全部取消。

5.3 符号矩阵的基本运算

几乎所有的符号函数都涉及到符号表达式和符号矩阵，并返回一个符号表达式或符号数组（即便是看起来像数字的数据，也是一个内部用字符串表达的符号表达式，这可以运用 MATLAB 的 isstr 函数检验之）。

5.3.1 四则运算

MATLAB 5.0 以上版本的符号矩阵的四则运算指令，形式上开始和数值计算的双精度数的运算完全相同，而不再是必须用专门的运算函数。即用数学运算符号+、一、*、.*、/、./、\、.\ 即可代表通常的加、减、乘、点乘、左除、左点除、右除以及右点除四则运算。同时可以通过单个运算符号的复合使用来完成较为复杂的符号矩阵运算。

【例 5-5】对两个符号矩阵作基本四则运算。

```
syms a b c d
A=sym('[a b;c d]')                %创建符号矩阵 A
A =
    [ a, b]
    [ c, d]
B=sym('[a+b,a-b;c+d,c-d]');       %创建符号矩阵 B
A+B
ans =
    [ 2*a+b,        a]
    [ 2*c+d,        c]
A-B
ans =
    [      -b, 2*b-a]
    [      -d, 2*d-c]
A*B
ans =
    [ a*(a+b)+b*(c+d), a*(a-b)+b*(c-d)]
    [ c*(a+b)+d*(c+d), c*(a-b)+d*(c-d)]
A.*B
ans =
    [ a*(a+b), b*(a-b)]
    [ c*(c+d), d*(c-d)]
A/B
ans =
    [  1/2*(-a*c+a*d+b*d+b*c)/(a*d-b*c),      1/2*(-b^2-2*a*b+a^2)/(a*d-b*c)]
    [      1/2*(-c^2+d^2+2*c*d)/(a*d-b*c), -1/2*(-a*c+a*d+b*d+b*c)/(a*d-b*c)]
```

```
A./B
ans =
    [ a/(a+b), b/(a-b)]
    [ c/(c+d), d/(c-d)]
A.\B
ans =
    [ (a+b)/a, (a-b)/b]
    [ (c+d)/c, (c-d)/d]
A^2                         %对矩阵A进行乘方运算
ans =
    [ a^2+b*c, a*b+b*d]
    [ a*c+c*d, b*c+d^2]
A.^2                        %对符号矩阵A中的每个符号元素进行乘方运算
ans =
    [ a^2, b^2]
    [ c^2, d^2]
    A+B+A^2
ans =
    [ 2*a+b+a^2+b*c,      a+a*b+b*d]
    [ 2*c+d+a*c+c*d,      c+b*c+d^2]
```

从例 5-5 可看出，符号矩阵的运算与数值矩阵一样方便和简单，并且与通常的书写规则相一致。

需要说明以下几点：

- MATLAB 5.0 以上版本采用了重载技术，对数学运算符运用了重载技术，因此对双精度运算的运算符同样适用于符号对象；
- 符号对象的加法和减法需要满足：两个符号矩阵的大小相等方可进行加减运算；符号矩阵可以和符号标量进行加减运算，运算按照数组运算规则进行；
- 两个符号矩阵只有在内积相等时才可以进行乘法运算；
- 符号的乘方运算 S^p，若 S 为符号表达式，p 可以为符号表达式或数值表达式；若 S 为符号矩阵，则 p 必须是整数。

5.3.2　符号矩阵线性代数运算

和数值矩阵一样，符号矩阵也可以进行矩阵的线性代数运算，例如，求一个符号矩阵行列式的值、求逆、求方阵的特征值和特征向量等。本节将通过例 5-6 来对符号矩阵的这类用法作一个简单的示范。

【例 5-6】对一个符号矩阵作线性代数运算。

```
format short
H=hilb(2)                   %采用短格式输出方式，创建一个2×2的hilbert矩阵
H =
```

```
     1.0000     0.5000
     0.5000     0.3333
H=sym(H)                    %将数值矩阵H转化为符号矩阵
H =
[   1, 1/2]
[ 1/2, 1/3]
inv(H)                      %求符号矩阵H的逆矩阵
ans =
[  4, -6]
[ -6, 12]
det(H)                      %求符号矩阵H的行列式值
ans =
1/12
```

本节仅仅是说明符号矩阵的此类运算，详细的应用和使用方法将在第 6 章进行阐述。

5.3.3 MATLAB 关于不同精度的控制

相对于 MATLAB 的数值算法和符号算法，在 MATLAB 所有的工具箱中的运算，共有三种针对不同精度的算法，分别为：

- 针对浮点运算的数值算法

它是计算速度最快，占用计算机内存最少的算法，它与 C 语言和 FORTRAN 语言中的浮点运算算法完全相同。由于针对浮点运算的数值算法在机器内部都是以若干位二进制数表达的，它在机器内的表达和计算都是一个被“截断”的近似值，因此数值算法速度最快，精度也最差。MATLAB 双精度输出的数字位数由 format 命令控制，但其内部表示法总是采用由计算机硬件提供的 8 位浮点表示法。

- 针对精确运算的符号算法

其计算时间最长，内存占用最多，精度也最高。只要有足够的内存和足够长的时间，其产生的结果也最精确。

- 任意精度的算法

其运算时间、内存占用和计算精度均介于以上两种运算之间。采用函数 digits 来控制十进制结果的有效位数。增加有效位数将增加结果的精度，同时也增加了计算时间和内存要求。digits 的缺省值为 32，大约对应于浮点精度。

符号数学工具箱中，可用 vpa 函数命令执行任意精度运算，其格式为：

- ◆ *R* = vpa(*S*)，计算 *S* 中的每个元素到任意位数，有效位数为函数 digits 指定的当前值，返回的结果 *R* 是一个符号对象；
- ◆ vpa(*S*,*D*)，用参数 *D*(Digits)指定有效数字位数，而不是用函数 digits 的设置。

5.4　可视化的符号函数分析界面

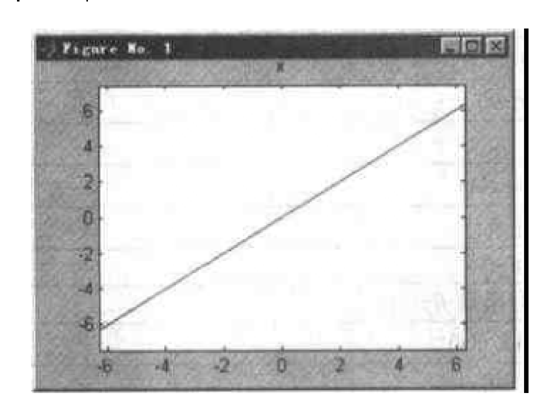
图形化的函数计算器是 MATLAB 符号数学 sym('arg2','unreal'); ...工具箱所提供的第三种符号计算方法，为符号函数可视化提供更为简便易用的命令。本节着重介绍两个进行数学分析的可视化界面：单变量函数分析界面和泰勒级数逼近分析界面。

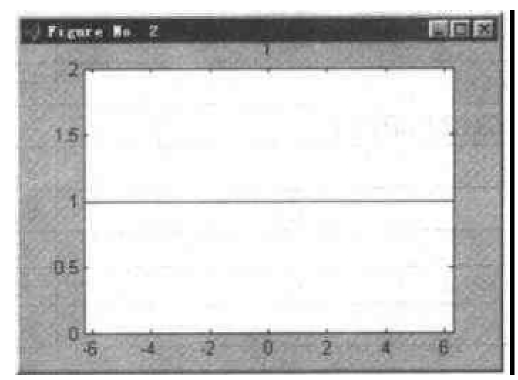
5.4.1　单变量函数分析界面

单变量函数分析界面用于考察两个一元函数各自性质及其相关关系。该函数计算器由 funtool.m 文件生成。在 MATLAB 命令窗口中键入下面命令即可：

funtool

该命令运行后，系统将产生三个如图 5-1 所示的新窗口。其中 1 号和 2 号窗口（Figure No.1，Figure No.2）是函数曲线窗口，3 号窗口（Figure No.3）是函数运算控制器。三个窗口中只有一个处于激活状态，可以用鼠标单击窗口，激活该视窗。





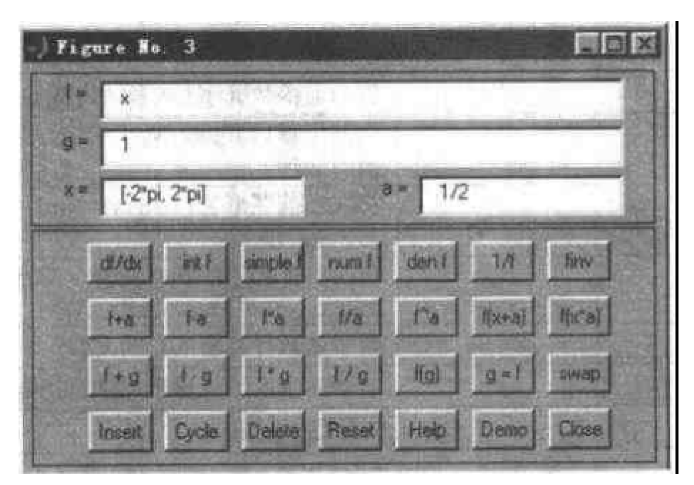


图 5-1　单变量函数分析窗口

函数运算控制器窗口（Figure No.3）上半部分的 f、g、x、a 分别是 Figure No.1 和 Figure No.2 中所示曲线的一元函数 $f(x)$、$g(x)$的定义式、自变量 x 的取值范围和常数 a 的值。用户可以自行设定这些量的值。控制窗口的下半部分是运算命令和辅助操作部分，共有四行，分别为单函数运算、函数和常数 a 的运算、两个函数之间的运算和辅助操作部分。各个部

分的详细含义如表 5-1 所示。

表 5-1 单变量函数分析窗口按键名称及含义

按键名称	功能和含义
df/dx	对函数 $f(x)$ 关于自变量 x 求导
int f	对函数 $f(x)$ 关于自变量 x 积分
simple f	对函数 $f(x)$ 化简
num f	取 $f(x)$ 的分子表达式
den f	取 $f(x)$ 的分母表达式
1/f	求 $1/f(x)$
finv	求 $f(x)$ 的反函数
函数和常数 a 的运算	
$f+a$	计算 $f(x)+a$
$f-a$	计算 $f(x)-a$
$f*a$	计算 $f(x)\times a$
f/a	计算 $f(x)/a$
f^a	计算 $f^a(x)$
$f(x+a)$	计算 $f(x+a)$
$f(x*a)$	计算 $f(ax)$
两个函数之间的运算	
$f+g$	求 $f(x)+g(x)$
$f-g$	求 $f(x)-g(x)$
f/g	求 $f(x)/g(x)$
$f(g)$	求符合函数 $f(g(x))$
$f*g$	求 $f(x)\times g(x)$
$g=f$	令 $g(x)=f(x)$
swap	交换 $f(x)$、$g(x)$ 的定义式
辅助操作部分	
Insert	将当前函数 $f(x)$ 插入系统内含的典型函数演示表中
Cycle	在图形窗口 figure No.1 中顺序显示典型函数演示表中的函数曲线
Delete	将在当前图形窗口 figure No.1 中的函数 $f(x)$ 从典型函数演示表中删除
Reset	将函数计算器的状态调整至初始状态： $f(x)=x$; $g(x)=1$; x=[-pi,pi];a=1/2
Help	查看函数计算器的帮助文件
Demo	自动演示函数计算器的计算功能
Close	关闭函数计算器

5.4.2 泰勒级数逼近分析界面

从 MATLAB 5.3 版本以上开始的符号数学工具箱 2.1 版新增加了泰勒级数逼近分析界面，用于观察函数 $f(x)$ 在给定区间位置上的被 N 阶泰勒多项式 $T_N(x)$ 逼近的情况。在 MATLAB 工作窗口中输入命令：

taylortool

即可产生如图 5-2 所示泰勒级数分析窗口。

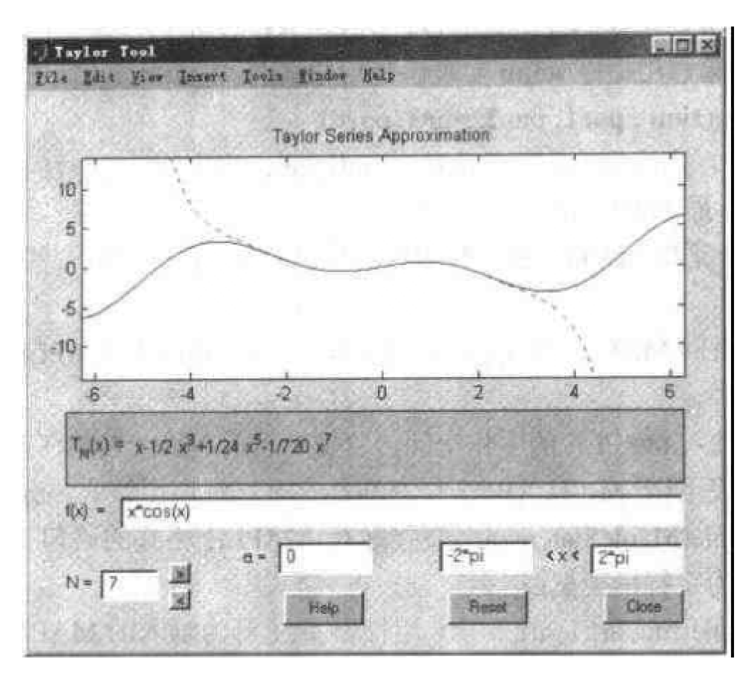


图 5-2　泰勒级数分析窗口

函数 $f(x)$有两种输入方式：

- 一开始直接由命令 taylortool(*fx*)引入，此处 *fx* 必须是字符串表达式；
- 在泰勒级数分析窗口的 $f(x)$栏中直接由键盘输入表达式，按 Enter 键确认。

N 表示泰勒级数的阶次，系统的缺省值为 7，可以通过右侧的按键改变阶次，也可以通过键盘直接输入。*a* 栏表示泰勒级数的展开点（即在 $x_0=a$ 处展开），系统缺省设置为 0。"<x<" 两侧为自变量 *x* 的取值范围，缺省设置为(-2 π ,2 π)，用户可以自行改变。

注意： *在窗口的各栏中进行修改后，必须按 Enter 键进行确认，方可生效。*

5.5　使用 MAPLE 的符号资源

5.5.1　MAPLE 与 MATLAB 的连接命令

MAPLE 具有强大的符号计算功能和丰富的经典应用数学函数，MATLAB 的符号数学工具箱（Symbolic Math Toolbox）就是以 MAPLE 的内核为"引擎"，依靠 MAPLE 已有的库函数（Library）开发的。由于 MAPLE 资源是以库的形式而不是以 M 文件的形式提供给 MATLAB，因此无法在 MATLAB 中直接使用。为了能够在 MATLAB 的工作环境中进一步利用 MAPLE 的符号计算能力，MATLAB 提供有专门的指令用于 MATLAB 和 MAPLE 的连接，如下所述：

- mfun　对 MAPLE 中的若干重要的特殊函数实施数值计算。对于 MAPLE 中的"完

全椭圆积分”、“正弦积分”以及“误差函数”等一些重要的特殊函数，由于在许多的工程应用中经常需要计算此类函数的值，并且在符号求积和微分方程符号解中也常常要计算此类函数的值，MATLAB 便将这些重要的特殊函数引入符号数学工具箱，用命令 mfun 实现。使用格式为：

mfun('function', par1, par2, par3, par4)

该函数以数值方式计算 MAPLE 中特殊函数'function'的值，函数的参数由 par1、par2、par3 和 par4 指定，最多可以指定四个参数。

- mfunlist　采用 MATLAB 注释语句列出能被 mfun 计算的一些重要 MAPLE 函数列表；
- mhelp　查阅 MAPLE 库函数的联机帮助文件，以获取 MAPLE 库函数及其调用方法；
- maple　进入 MAPLE 的工作空间，直接对访问 MAPLE 的任意函数进行计算，并将结果返回至 MATLAB 工作空间。它有三种调用格式，分别为：
  - ◆ *R*=maple(MapleStatement) 直接运行 MAPLE 格式的语句 MapleStatement，输出 *R* 为字符串类型；
  - ◆ *R*=maple(fun, arg1, arg2, ...) 运行以 arg1 等为输入的 MAPLE 的 fun 函数；
  - ◆ [*R*, *S*]=maple(...) 将 MAPLE 运行结果由输出总量返回。

【例 5-7】调用 MAPLE 函数，求解 $\sin(x^2+y^2)$在 $x=0$，$y=0$ 处展开的截断八阶小量的泰勒级数近似式。

调用格式一：

```
tl1=maple('mtaylor(sin(x^2+y^2),[x=0,y=0],8)')
tl1 =
mtaylor(sin(x+y),[x = 0, y = 0],8)
```

直接用 maple 调用 mtaylor 函数，输出结果是输入指令本身，说明该调用函数还没有被读入。此外，用户可以从 Mhelp mtaylor 在线帮助中看到，运行 mtaylor 之前必须先用 maple('readlib(function)')命令读入 MAPLE 的库函数。

```
maple('readlib(mtaylor)');
tl1=maple('mtaylor(sin(x^2+y^2),[x=0,y=0],8)')
tl1 =
x^2+y^2-1/6*x^6-1/2*y^2*x^4-1/2*y^4*x^2-1/6*y^6
```

调用格式二：

```
maple('readlib(mtaylor)');
tl2=maple('mtaylor','sin(x^2+y^2)','[x=0,y=0]','8')
tl2 =
x^2+y^2-1/6*x^6-1/2*y^2*x^4-1/2*y^4*x^2-1/6*y^6
```

从例 5-7 可以看出，两种调用方法所得的结果相同。

- procread　把按照 MAPLE 格式写的源程序读入 MAPLE 工作空间。对于需要运行由多个 MAPLE 指令构成的程序时，必须由 procread 和 maple 相结合方可解决。先用 maple 格式写好源程序，再用 procread 命令将此程序读入 MAPLE 工作空间。

关于 maple 编程特征请读者参阅 MAPLE 手册。

5.5.2 MAPLE 特殊函数清单及其调用

为了方便用户查阅函数，MATLAB 编制了一个纯说明文件 mfunlist.m，用于列出 MAPLE 的几十条常用的重要函数名以及简单的说明。用户可直接在 MATLAB 工作窗口中键入 mfunlist，即可显示出这些特殊函数及其所需参数。例如，在工作窗口中键入：

mfunlist

在屏幕上将显示如下内容：

```
MFUNLIST Special functions for MFUN.
   The following special functions are listed in alphabetical order
   according to the third column. n denotes an integer argument,
   x denotes a real argument, and z denotes a complex argument. For
   more detailed descriptions of the functions, including any
   argument restrictions, see the Reference Manual, or use MHELP.

  Bernoulli       n        Eernoulli Numbers
  Bernoulli       n,z      Bernoulli Polynomials
  BesselI         x1,x     Bessel Function of the First Kind
  BesselJ         x1,x     Bessel Function of the First Kind
  BesselK         x1,x     Bessel Function of the Second Kind
  BesselY         x1,x     Bessel Function of the Second Kind
  Beta            z1,z2    Beta Function
  Binomial        x1,x2    Binomial Coefficients
  LegendreKc      x        Complete Elliptic Integral of First Kind
  LegendreEc      x        Complete Elliptic Integral of Second Kind
  LegendrePic     x1,x     Complete Elliptic Integral of Third Kind
  LegendreKc1     x        LegendreKc using Complementary Modulus
  LegendreEc1     x        LegendreEc using Complementary Modulus
  LegendrePic1    x1,x     LegendrePic using Complementary Modulus
  Erfc            z        Complementary Error Function
  Erfc            n,z      Complementary Error Function's Iterated Integrals
  Ci              z        Cosine Integral
  Dawson          x        Dawson's Integral
  Psi             z        Digamma Function
  Dilog           x        Dilogarithm Integral
  Erf             z        Error Function
  Euler           n        Euler Numbers
  Euler           n,z      Euler Polynomials
  Ei              x        Exponential Integral
```

Ei	n,z	Exponential Integral
FresnelC	x	Fresnel Cosine Integral
FresnelS	x	Fresnel Sine Integral
GAMMA	z	Gamma Function
Harmonic	n	Harmonic Function
Chi	z	Hyperbolic Cosine Integral
Shi	z	Hyperbolic Sine Integral
Hypergeom	X1,X2	(Generalized) Hypergeometric Function
LegendreF	x,x1	Incomplete Elliptic Integral of First Kind
LegendreE	x,x1	Incomplete Elliptic Integral of Second Kind
LegendrePi	x,x2,x1	Incomplete Elliptic Integral of Third Kind
GAMMA	z1,z2	Incomplete Gamma Function
W	z	Lambert's W Function
W	n,z	Lambert's W Function
LnGAMMA	z	Logarithm of the Gamma function
Li	x	Logarithmic Integral
Psi	n,z	Polygamma Function
Ssi	z	Shifted Sine Integral
Si	z	Sine Integral
Zeta	z	(Riemann) Zeta Function
Zeta	n,z	(Riemann) Zeta Function
Zeta	n,z,x	(Riemann) Zeta Function

Orthogonal Polynomials (Extended Symbolic Math Toolbox only)

T	n,x	Chebyshev of the First Kind
U	n,x	Chebyshev of the Second Kind
G	n,x1,x	Gegenbauer
H	n,x	Hermite
P	n,x1,x2,x	Jacobi
L	n,x	Laguerre
L	n,x1,x	Generalized Laguerre
P	n,x	Legendre

以上函数中，用 maple 命令只能调用少数几个，而 mfun 是调用它们的专门命令。

5.6 小　　结

MATLAB 的符号计算功能和符号工具箱使 MATLAB 语言真正成为“演算纸式的计算语言”，几乎涵盖了所有的数学计算内容。本章初步介绍了 MATLAB 的符号计算功能，较为深入地学习及示例将在第 6 章给予详细说明。通过本章的学习，应掌握如下内容：

（1）了解 MATLAB 6.0 的符号变量，掌握 MATLAB 符号表达式、符号矩阵的两种创建方法，以及符号矩阵的转换、引用和修改。

（2）掌握 MATLAB 符号数学函数的创建以及如何在数值型、字符型和符号型三种数据类型之间相互转换。

（3）熟练掌握符号矩阵的基本运算及 MATLAB 关于不同精度的控制方法。

（4）学会使用 MATLAB 可视化的符号函数分析界面，包括单变量函数和泰勒级数逼近分析界面的使用。

（5）初步了解 MAPLE 的符号资源，掌握 MAPLE 与 MATLAB 的连接命令，以及如何调用 MAPLE 的特殊函数。

习　　题

1．建立一个包含 2sin(x)cos(x)的符号表达式，并将该表达式转换为符号变量。

2．用符号计算验证三角恒等式 $\sin\varphi_1\cos\varphi_2-\cos\varphi_1\sin\varphi_2=\sin(\varphi_1-\varphi_2)$ 的正确性。

3．用 rand 函数建立一个 3×3 维随机数值矩阵，并将该数值矩阵转换为符号矩阵，比较它们的不同。

4．分别用 1 位和 10 位有效数字计算 2*pi+3.1415 的值，并比较与采用 format 格式的差别。

5．求矩阵 $A=\begin{bmatrix}a_{11} & a_{12} & a_{13}\\ a_{21} & a_{22} & a_{23}\\ a_{31} & a_{32} & a_{33}\end{bmatrix}$ 的行列式值、并求出矩阵 $B=\begin{bmatrix}a_{11} & a_{12}\\ a_{21} & a_{22}\end{bmatrix}$ 的特征根。

6．多次运用 simplify 命令简化 $f=\cos x+\sqrt{-\sin^2 x}$，并与采用 simple 命令得出的结果相比较。

7．利用 MAPLE 特殊函数库中的 bernoulli 函数，求解 x 的 6 阶贝努利表达式。

第 6 章　MATLAB 在工程教学中的应用

本章作为本书的核心部分，从工程教学的角度，详细并系统地介绍了 MATLAB 在高等数学、线性代数、符号运算和数据处理等方面的应用，同时给出大量的实例，并结合图形图像处理和数据可视化等 MATLAB 的图像功能进行讲述。通过本章的学习，可熟练掌握 MATLAB 关于数学计算方面的内容。

需要说明的一点是，本章是本着解决问题的目的进行编写的，因此不再区分此解决方法是属于数值计算还是属于符号计算，无论采取何种计算方法，都是解决该类问题的一种选择。在确实需要区分时，会予以说明。

6.1　解线性方程组

在工程教学和实践中，对线性方程组的求解是一个重要的内容。在实际工程应用、实验数据分析以及进行理论研究等许多情况下，很多问题都可归结为线性方程组的求解。因而，线性方程组求解的应用非常广泛。求解线性方程组不可避免地要用到矩阵分解的概念，矩阵分解函数在第 2 章已有简单介绍，本节首先介绍用于解线性方程组的几个矩阵分解函数，然后具体讲解各类线性方程组的解法。

6.1.1　矩阵的分解

矩阵分解是一个非常重要的概念，如在求方程组的解、矩阵的特征值、特征向量和矩阵的秩等重要参数时都要用到矩阵的分解。在实际工程中，某些场合也要对矩阵进行特定形式的分解，总之，无论在理论证明或是科学计算上，对矩阵进行分解操作都十分重要。MATLAB 提供了许多矩阵分解函数，利用这些函数可以比较容易地进行矩阵分解。下面着重介绍其中几种矩阵分解的方法，其相关函数如表 6-1 所示。

表 6-1　　矩阵分解函数

函数命令	功能	说明
chol	cholesky 分解	对称正定矩阵的分解，用于求解方程组
lu	lu 分解	矩阵采用 lu 分解，可直接求解线性方程组
qr	正交三角分解	把矩阵分解为正交矩阵或酉矩阵和三角矩阵
svd	奇异值分解	把矩阵分解为一个对角阵和两个酉矩阵
schur	schur 分解	把矩阵分解为一个 schur 矩阵和一个酉矩阵

在 MATLAB 中，线性方程组的求解主要用到三种基本的矩阵分解，即对称正定矩阵的 cholesky 分解、一般方程的 gaussian 消去法和矩阵的正交分解。这三种分解由函数 chol、lu 和 qr 完成。分解时都使用三角矩阵，其中所有位于对角线以上或以下的元素都为 0。

矩阵经分解成为三角矩阵后，线性方程组不论是用左除还是右除都可以简单、快速地求解。

1. 对称正定矩阵的 cholesky 分解

并不是所有的对称矩阵都可以进行 cholesky 分解，能进行此种分解的矩阵必须是正定的，即矩阵 X 的所有对角线元素都是正的，而非对角线元素不会太大。此种分解可用于复矩阵，此时复矩阵必须满足 $X'=X$，是 hermitian 正定的。

设 X 是一个 $n\times n$ 的对称正定矩阵，则存在对角线为正的上三角矩阵 R，使 $X=R'R$。

在 MATLAB 中，cholesky 分解的函数调用格式有两种：

- R=chol(X) X 是对称正定矩阵，R 是上三角矩阵，使 $R'R=X$。如果矩阵 X 是非正定的，则会给出错误信息；
- [R,p]=chol(X) 返回两个参数，不会给出出错信息。如果 X 是正定的，则 P 等于 0，R 同上；如果 X 是非正定的，则 P 等于正整数，R 是一个阶数为 $q=p$-1 的上三角阵，使得 $R'*R = X(1:q,1:q)$。

下面通过例 6-1 来说明其用法：

【例 6-1】cholesky 分解。

```
X=[2 2 -2; 2 5 -4;-2 -4 5]        %输入对称正定矩阵 X
R=chol(X)                         %进行 cholesky 分解

X =
     2     2    -2
     2     5    -4
    -2    -4     5
R =
    1.4142    1.4142   -1.4142
         0    1.7321   -1.1547
         0         0    1.2910
R'*R
ans =
    2.0000    2.0000   -2.0000
    2.0000    5.0000   -4.0000
   -2.0000   -4.0000    5.0000
```

2. lu 分解

矩阵的 lu 分解在工程教学和实践中非常重要，许多运算都以 lu 分解为基础。如方阵的求逆操作 inv()、行列式求值 det()以及解线性方程组等。lu 分解是除法运算的基础。gaussian 消去法或 lu 分解是将任何方阵表示为一个下三角矩阵 L 和一个上三角矩阵 U 的乘积，其调用格式有两种：

- [L,U] = lu(X) 此函数命令将得到一个上三角矩阵并且存储在 U 中和一个准下三角矩阵并且存储在 L 中，使得 $X=LU$。准下三角矩阵 L 实际上是下三角矩阵的转置矩阵；
- [L,U,P] = lu(X) 此种调用格式将得到一个主对角元为 1 的下三角矩阵 L、上三角

矩阵 U 和一个由 0 和 1 组成的置换矩阵 P，使得 $PX=LU$。

注意：进行 lu 分解时，矩阵 X 必须是方阵。

在 MATLAB 中，lu 分解允许线性方程组 $Ax=b$ 进行如下快速运算：

x=U\(L\b)

矩阵行列式的值和矩阵求逆也可以利用 lu 分解进行如下计算：

det(X)=det(L)*det(U)

inv(X)=inv(U)*inv(L)

下面举例 6-2 来说明如何进行 lu 分解。

【例 6-2】lu 分解。

```
X=[3 -1 2;1 2 -1; -2 1 4]                %输入矩阵 X
X =
     3    -1     2
     1     2    -1
    -2     1     4
[L,U,P]=lu(X)                            %进行 lu 分解
L =
    1.0000         0         0
    0.3333    1.0000         0
   -0.6667    0.1429    1.0000
U =
    3.0000   -1.0000    2.0000
         0    2.3333   -1.6667
         0         0    5.5714
P =
     1     0     0
     0     1     0
     0     0     1
P*X
ans =
     3    -1     2
     1     2    -1
    -2     1     4
L*U
ans =
    3.0000   -1.0000    2.0000
    1.0000    2.0000   -1.0000
   -2.0000    1.0000    4.0000
```

3. qr 分解

qr 分解即矩阵的正交分解，将矩阵分解为一个正交矩阵和一个上三角矩阵的乘积。

此种分解适用于任意矩阵，是非常重要的分解形式。其调用格式为：

- [*Q*,*R*] = qr(*X*)　此种调用格式生成一个与 *X* 同阶的上三角矩阵 *R* 和一个正交矩阵 *Q*，使得 *X*=*QR*；
- [*Q*,*R*,*E*] = qr(*X*)　此种格式将得到一个置换矩阵 *E*、上三角矩阵 *R* 和正交矩阵 *Q*，使得 *XE* = *QR*，选择置换矩阵 *E* 使得 abs(diag(*R*))递减；
- [*Q*,*R*] = qr(*X*,0)　将生成一种"经济"的分解。如果矩阵 *X* 是 $m\times n$ 阶，并且 $m>n$，则仅计算出 *Q* 的前 *n* 列；
- [*Q*,*R*,*E*] = qr(*X*,0)　生成一种"经济"的分解，其中 *E* 是一个置换向量，使得 *QR* = *A*(:,*E*)，选择列置换向量使得 abs(diag(*R*))递减。

【例 6-3】qr 分解。

```
X=[3 -1 2 5;1 2 -1 7; -2 1 -2 4]     %输入矩阵X
X =
     3    -1     2     5
     1     2    -1     7
    -2     1    -2     4
[Q,R]=qr(X)                          %进行qr分解
Q =
   -0.8018    0.1543    0.5774
   -0.2673   -0.9567   -0.1155
    0.5345   -0.2469    0.8083
R =
   -3.7417    0.8018   -2.4054   -3.7417
         0   -2.3146    1.7591   -6.9128
         0         0   -0.3464    5.3116
[Q,R,E]=qr(X,0)
Q =
   -0.5270    0.6463    0.5518
   -0.7379    0.0259   -0.6745
   -0.4216   -0.7626    0.4905
R =
   -9.4868   -1.4757   -1.3703    0.5270
         0    3.4383   -1.4607    2.8437
         0         0   -1.4102    0.7971
E =
     4     1     2     3
```

4. 奇异值分解

奇异值分解在矩阵分析中占有极为重要的位置。奇异值分解的定义为：

对于任意的矩阵 $A\in C^{m\times n}$，存在酉矩阵（Unitary matrix），$U=[u_1, u_2, \ldots, u_m]$，$V=[v_1, v_2, \ldots, v_n]$ 使得：

U^TAV=diag(σ_1, σ_2,…, σ_p)　其中，$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_p \geq 0$，$p$=min{$m$，$n$}

上式中，{σ_i，u_i，v_i}分别称为矩阵 A 的第 i 个奇异值、左奇异向量和右奇异向量，而它们的组合就称为奇异值分解三对组。这里的上标“T”表示共轭转置。

MATLAB 中提供的奇异值分解函数 svd 的调用格式为：

- [U,S,V] = svd(X)产生一个与矩阵 X 具有相同维数的矩阵 S，其对角线元素为递减的非负值，同时得到酉矩阵 U 和 V，使得 $X = U$*S*V'；
- S = svd(X) 得到矩阵 X 的奇异值组成的向量；
- [U,S,V] = svd(X,0) 得到一个“经济大小”的分解结果，如果 X 是 $m \times n$ 矩阵且 $m > n$，则只计算 U 矩阵的前 n 行，且 S 矩阵是 $n \times n$ 阶的。

【例 6-4】奇异值分解。

```
X=[3 1 4;2 2 4;1 -3 -2;1 2 3]      %输入矩阵 X
X =
     3     1     4
     2     2     4
     1    -3    -2
     1     2     3
[U,S,V]=svd(X)                      %进行奇异值分解
U =
     0.5873   -0.5075   -0.5954    0.2073
     0.5951   -0.0691    0.4057   -0.6903
    -0.3132   -0.8424    0.4331    0.0688
     0.4503    0.1674    0.5417    0.6898
S =
    8.2230         0         0
         0    3.2221         0
         0         0         0
         0         0         0
V =
    0.3757   -0.7249    0.5774
    0.4400    0.6878    0.5774
    0.8157   -0.0371   -0.5774
[U,S,V]=svd(X,0)                    %进行奇异值分解
U =
    0.5873   -0.5075   -0.5954
    0.5951   -0.0691    0.4057
   -0.3132   -0.8424    0.4331
    0.4503    0.1674    0.5417
S =
    8.2230         0         0
```

```
        0    3.2221         0
        0         0         0
V =
   0.3757   -0.7249   0.5774
   0.4400    0.6878   0.5774
   0.8157   -0.0371  -0.5774
```

奇异值分解也是矩阵求秩运算的基础，对矩阵 A 进行奇异值分解 S=svd(A)，得到向量 s 的非零元素的个数就是矩阵 A 的秩。如：

```
X=[3 1 4;2 2 4;1 -3 -2;1 2 3] ;
s=svd(X)
s =
   8.2230
   3.2221
        0
```

可见矩阵 X 的秩为 2，用求秩运算 rank(X)可以验证这一结果。

5. schur 分解

schur 分解的定义为：

$$A=USU^T$$

其中 A 是方阵，U 是一个正交矩阵，S 是一个块上三角矩阵，由对角线上的 1×1 和 2×2 块组成。特征值由 S 的对角线和块给出，而 U 的列比一系列特征向量给出了更多的数值特性。schur 分解可以对缺陷矩阵进行。

矩阵的复 schur 形式矩阵是一个特征值在对角线上的上三角阵；而实 schur 形式是实特征值在对角线上，复特征值以 2×2 的块矩阵排列在对角线上。

用函数 T = schur(X)求矩阵 X 的 schur 形式矩阵时，如果 X 是实矩阵，函数返回实数 schur 形式矩阵；如果 X 是复矩阵，函数将返回复数 schur 形式矩阵。命令函数 rsf2csf()可以将实数形式转化为复数形式。矩阵 X 必须是方阵。

函数 schur 的调用格式为：

- T = schur(X)　仅仅返回 schur 形式矩阵 T；
- [U,T] = schur(X)　得到 schur 形式矩阵 T 和酉矩阵 U，使得 $X = U*T*U'$ 和 $U'*U$ = eye(size(X))。

【例 6-5】schur 分解。

```
A=[1 2 -1;3 4 0;1 2 3]      %输入方阵 A
A =
     1     2    -1
     3     4     0
     1     2     3
[U,T]=schur(A)              %进行 schur 分解
U =
    0.8262   -0.2294   -0.5145
```

```
     -0.5571    -0.4680    -0.6860
      0.0835    -0.8534     0.5145
T =
     -0.4495     0.8575     0.3760
     -0.0000     4.4495     2.8501
           0          0     4.0000
[UU,TT]=rsf2csf(U,T)                    %对 U 和 T 进行转换
UU =
      0.8262    -0.2294    -0.5145
     -0.5571    -0.4680    -0.6860
      0.0835    -0.8534     0.5145
TT =
     -0.4495     0.8575     0.3760
           0     4.4495     2.8501
           0          0     4.0000
```

6. 矩阵的 hessenberg 分解

矩阵的 hessenberg 分解由函数 bess 完成，该函数命令的调用格式为：

- *H* = hess(*A*)　可求矩阵的 hessenberg 形式矩阵 *H*，*H* 的第一子对角线以下的元素为 0。如果矩阵 *A* 是对称的或是 hermitian 矩阵，则 *H* 是对角三角阵，矩阵 *A* 必须是方阵；
- [*P*,*H*] = hess(*A*)　得到一个酉矩阵 *P* 和一个 bessenberg 形式矩阵 *H*，使得 *A* = *P***H***P*' 和 *P*'**P* = eye(size(*A*))。

【例 6-6】hessenberg 分解。

```
A=[1 2 -1;3 4 0;1 2 3]        %输入方阵 A
A =
     1     2    -1
     3     4     0
     1     2     3
[P,H]=hess(A)                 %进行 hessenberg 分解
P =
      1.0000          0          0
           0    -0.9487    -0.3162
           0    -0.3162     0.9487
H =
      1.0000    -1.5811    -1.5811
     -3.1623     4.5000     0.5000
           0    -1.5000     2.5000
```

6.1.2　线性方程组的求解

在工程教学和计算中，线性方程组的求解是一个很重要的问题。在矩阵表示方法中，线性方程组可以表示为给定两个矩阵 *A* 和 *B*，求 *X* 的惟一解，使得：

AX=B 或 XA=B

在 MATLAB 中，求解线性方程组时，主要采用前面章节介绍的除法运算符"/"和"\"求解。如：

- *X=A\B*　表示求矩阵方程 *AX=B* 的解；
- *X=B/A*　表示求矩阵方程 *XA=B* 的解。

对方程 *X=A\B*，要求矩阵 *A* 和 *B* 用相同的行数，*X* 和 *B* 有相同的列数，它的行数等于矩阵 *A* 的列数，方程 *X=B/A* 同理。

如果矩阵 *A* 不是方阵，其维数是 $m \times n$，则有：

- $m=n$　恰定方程，寻求精确解；
- $m>n$　超定方程，寻求最小二乘解；
- $m<n$　不定方程，寻求基本解，其中至多有 *m* 个非零元素。

针对不同的情况，MATLAB 将采用不同的算法来求解。

6.1.3　恰定方程组

恰定方程组由 *n* 个未知数的 *n* 个方程构成，方程有惟一的一组解，其一般形式可用矩阵、向量写成如下形式：

$Ax=b$

其中，*A* 是方阵，*b* 是一个列向量。

在线性代数教科书中，最常用的方程解法有：

- 利用 cramer 公式求解法；
- 利用矩阵求逆解法，即 $x=A^{-1}b$；
- 利用 gaussian 消去法；
- 利用 lu 法求解。

一般来说，对于维数不高、条件数不大的矩阵，上面四种解法所得的结果差别不大。前三种解法的真正意义是在其理论上，而不是实际的数值计算。在 MATLAB 中，出于对算法稳定性的考虑，行列式及逆的计算大都在 lu 分解的基础上进行。

在 MATLAB 中，求解这类方程组的命令十分简单，直接采用表达式：*x=A\b*。

MATLAB 的指令解释器在确认变量 *A* 非奇异后，就对它进行 lu 分解，并最终给出解 *x*；若矩阵 *A* 的条件数很大，MATLAB 会提醒用户注意所得解的可靠性。

如果矩阵 *A* 是奇异的，则 *Ax=b* 的解不存在，或者存在但不惟一；如果矩阵 *A* 接近奇异时，MATLAB 将给出警告信息；如果发现 *A* 是奇异的，则计算结果为 inf，并且给出警告信息；如果矩阵 *A* 是病态矩阵，也会给出警告信息。

注意：在求解方程时，尽量不要用 inv(*A*)**b* 命令，而应采用 *A\b* 的解法。因为后者的计算速度比前者快、精度高，尤其当矩阵 *A* 的维数比较大时。另外，除法命令的适用性较强，对于非方阵的 *A*，也能给出最小二乘解。

【例6-7】“左除”法与“求逆”法解恰定方程组时的性能比较。

```
rand('state',12);          %选定随机种子，产生随机矩阵 A
A=rand(100,100)+1.e8;      %生成（100×100）均匀分布的随机阵
                           %加大数值是要使矩阵 A 的条件数增大
x=ones(100,1);             %生成 100 行 1 列的向量
b=A*x;
flops(0);                  %浮点运算计数器置 0
tic                        %启动计时器 StopwatchTimer，开始计时
y=inv(A)*b;                %求逆法解方程运算次数
ti=toc                     %关闭计时器，并显示解方程所用的时间
ci=flops                   %求逆法解方程运算次数
erri=norm(y-x)             %结果与精确解的范-2 误差
errif=norm(A*y-b)          %方程的范-2 误差
%%%
flops(0);
tic
y=A\b;                     %用除法解方程
td=toc                     %关闭计时器，并显示解方程所用的时间
cd=flops                   %除法解方程运算次数
errd=norm(y-x)             %结果与精确解的范-2 误差
errdf=norm(A*y-b)          %方程的范-2 误差
```

运算结果：

```
ti =
    0.0600                 %求逆法解方程所用的时间
ci =
     2070322               %求逆法解方程所用的运算次数
erri =
     3.0708e-004
errif =
     6.6280e+004
td =                       %除法解方程所用的时间
    0
cd =                       %除法解方程所用的运算次数
    741872
errd =
     3.2243e-004
errdf =
     2.0095e-005
```

从本例执行的结果可知求逆法解方程所需运算次数是除法解方程的 2.5 倍，时间上除

法几乎是“机器零”。

6.1.4 超定方程组

对于方程组 $Ax=b$，A 为 $n\times m$ 矩阵，如果 A 列满秩，且 $n>m$，则方程没有精确解，此时称方程组为超定方程组。线性超定方程经常遇到的问题是数据的曲线拟合。对于超定方程，在 MATLAB 中，利用左除命令（$x=A\backslash b$）来寻求它的最小二乘解；还可以用广义逆来求，即 $x=\text{pinv}(A)$，所得的解不一定满足 $Ax=b$，x 只是最小二乘意义上的解。左除的方法是建立在奇异值分解基础之上，由此获得的解最可靠；广义逆法是建立在对原超定方程直接进行 householder 变换的基础上，其算法可靠性稍逊于奇异值分解，但速度较快。

【例 6-8】求超定方程组 $\begin{cases}2x_1-x_2+3x_3=3\\3x_1+x_2-5x_3=0\\4x_1-x_2+x_3=3\\x_1+3x_2-13x_3=-6\end{cases}$ 的解。

```
A=[2 -1 3;3 1 -5;4 -1 1;1 3 -13]      %输入矩阵A
A =
     2    -1     3
     3     1    -5
     4    -1     1
     1     3   -13
b=[3 0 3 -6]';
rank(A)
ans =
     3
x1=A\b            %左除解方程
x2=pinv(A) b    %广义逆求解
x1 =
    1.0000
    2.0000
    1.0000
x2 =
    1.0000
    2.0000
    1.0000
A*x1-b
ans =
   1.0e-014 *
    -0.0888
    -0.0888
```

```
    -0.1332
          0
```

可见 $x1$ 并不是方程 $Ax=b$ 的精确解，用 $x2$=pinv(A)*b 所得的解与 $x1$ 相同。

6.1.5 欠定方程组

欠定方程组未知量个数多于方程个数，但理论上有无穷个解。MATLAB 将寻求一个基本解，其中最多只能有 m 个非零元素。特解由列主元 qr 分解求得。

【例 6-9】解欠定方程 $\begin{cases} x_1-2x_2+x_3+x_4=1 \\ x_1-2x_2+1x_3-x_4=-1 \\ x_1-2x_2+x_3+5x_4=5 \end{cases}$

```
A=[1 -2 1 1;1 -2 1 -1;1 -2 1 5]         %输入矩阵 A
A =
     1    -2     1     1
     1    -2     1    -1
     1    -2     1     5
b=[1 -1 5]';
x1=A\b                                   %左除法解方程
Warning: Rank deficient, rank = 2   tol = 4.6151e-015
x1 =
          0
    -0.0000
          0
     1.0000
x2=pinv(A)*b                             %用广义逆解方程
x2 =
          0
    -0.0000
     0.0000
     1.0000
```

6.1.6 方程组的非负最小二乘解

在某种情况下，所求的线性方程组的解出现负数是没有意义的。虽然方程组可以得到精确解，但却不能取负值解。在这种情况下，其非负最小二乘解比方程的精确解更有意义。在 MATLAB 中，求非负最小二乘通常用函数 nnls，其调用格式为：

- X=nnls(A,b)　返回方程 $Ax=b$ 的最小二乘解，方程的求解过程被限制在 $x\geqslant 0$ 的条件下；
- X=nnls(A,b,TOL)　指定误差 TOL 来求解，TOL 的默认值为 TOL = max(size(A)) * norm(A,1) * eps，矩阵的-1 范数越大，求解的误差越大；

- [X,W] = nnls(A,b)　当 $x(i) = 0$ 时，$w(i) < 0$；当 $x(i) > 0$ 时，$w(i)\approx 0$，同时返回一个双向量 w。

【例 6-10】求方程组的非负最小二乘解。

```
A=[3.4336    -0.5238     0.6710
   -0.5238    3.2833    -0.7302
    0.6710   -0.7302     4.0261];       %输入矩阵A
b=[-1.0000    1.5000     2.5000 ];
[X,W]=nnls(A,b)                          %求方程的非负最小二乘解
X =
         0
    0.6563
    0.6998
W =
   -3.6820
   -0.0000
   -0.0000
x1=A\b               %用除法解方程
x1 =
   -0.3569
    0.5744
    0.7846
A*X-b                %验证非负最小二乘解
ans =
    1.1258
    0.1437
   -0.1616
A*x1-b               %验证除法解
ans =
    1.0e-015 *
   -0.2220
    0.4441
         0
```

6.1.7　方程解的精度

当人们用计算机计算一个问题时，最关心的是该问题的数值解。而所得的解是否可靠。除非计算所用的是一些特殊的整数或有理数，否则计算中的圆整误差、截断误差等都不可避免。

在 MATLAB 中，数与数之间的最小分辨率用 eps 表示。表达式中任何数的相对误差都不可能小于 eps。此外，在程序执行过程中，都不可避免地使这个最小误差比放大。范

数和条件数对求解过程中误差放大现象有重要的定量描述，在这里仅重点说明方程的精度问题。

在数值分析中，方程精度的估计可利用解的相对误差来进行，其公式为：

$$\frac{1}{K}\left(\frac{|\delta b|}{|b|}\right) \le \left(\frac{|\delta x|}{|x|}\right) \le K\left(\frac{|\delta b|}{|b|}\right)$$

其中，K 是矩阵的条件数，在 MATLAB 中的命令为 cond()。由于 eps 是机器精度，所以可以用 K*eps 的大小粗略判断所得的方程解是否可靠。

【例 6-11】对方程 $Ax=b$ 的近似解和精确解进行比较，其中 A 是 Hilbert 矩阵。

```
N=[6,14];                    %计算的矩阵阶数
for i=1:length(N)
    n=N(i);                  %所要计算的矩阵阶数放入 n 中
    A=hilb(n);               %产生 n 阶 Hilbert 矩阵
    Ai=invhilb(n);           %产生完全准确的 n 阶逆 Hilbert 矩阵
    b=ones(n,1);             %产生 n 阶全是 1 的向量
    x_app=A\b;               %利用左除来求近似解
    x_exa=Ai*b;              %利用准确求逆来求方程的准确解
fdb=norm(A*x_app-b);
fb=norm(b);
fdx=norm(x_app-x_exa);
fx=norm(x_app);
err_actual(i)=fdx/fx;        %实际的相对误差
K=cond(A);
err_app(i)=K*eps;            %最大可能的近似相对误差
err_max(i)=K*fdb/fb;         %最大可能的相对误差
end
disp('Hilbert 矩阵阶数'),disp(N)
format short e
disp('实际的相对误差'),disp(err_actual),disp('')
disp('最大可能的近似相对误差'),disp(err_app),disp('')
disp('最大可能的相对误差'),disp(err_max),disp('')
```

运行结果如下：

```
Hilbert 矩阵阶数
    6    14
实际的相对误差
  5.0339e-011   3.9641e+000
最大可能的近似相对误差
  3.3198e-009   9.4718e+001
最大可能的相对误差
  6.0095e-007   6.6239e+009
```

从该例可以看出，14阶的Hilbert矩阵是严重错误的。

6.1.8 用函数零点求方程的解

在MATLAB中，用函数零点法求方程或方程组的解有两个函数命令，即fzero和fsolve。在此，首先需对方程和方程组转化，比如将方程 $f(x)=g(x)$ 转化为 $F(x)=f(x)-g(x)=0$，方程组也是如此，然后再将函数 $F(x)$ 写成MATLAB的m函数，以便在fzeo和fsolve命令中调用。本小节将重点讲述其使用方法。

1. 一元方程转化的函数零点求法

在一元方程中，任意函数方程 $F(x)=0$ 可能有零点，也可能没零点；可能有一个零点，也可能有多个零点，很难有一个通用的解法。一般来说，其求解的过程为：先猜测一个初始零点，或者该零点大概所在的区间，然后通过计算，使猜测值不断精确化，或使猜测区间不断收缩，直到达到预先指定的精度为止。

在猜测一个初始的零点时，一般用MATLAB的作图命令来获取初始近似解。其具体步骤为：先确定一个零点可能存在的自变量区间，然后利用fplot命令绘出 $f(x)$ 在该区间的图形，用眼观察 $F(x)$ 与横轴的交点坐标，或者更细些，用zoom命令对交点局部放大来读数，或借助ginput命令来获得更精确的交点坐标。

● **利用fzero函数求一元函数零点**

命令fzero的调用格式为：

- ◆ x = fzero(fun,x0)　求一元函数零点命令的最简形式；
- ◆ [x,fval,exitflag] =fzero(fun,x0,options,P1,P2,...)　从MATLAB 5.3版本起，求一元函数零点命令最完整的格式。

其中参数 x0 是初始猜测的零点，同时也预定搜索零点的大致位置。它可以是标量或是二元向量，当 x0 是标量时，该命令将在它的两侧寻找一个与之最靠近的零点；当 x0 是一个二元向量[a，b]时，该命令将在区间[a，b]内寻找零点。

options是优化迭代所采用的参数选项。它是MATLAB设计优化程序fzero、fsolve、fminbnd、fminsearch和fminunc时都需要的一个"模块"，采用结构数组存放优化参数。如果没有优化参数要设置，可以在options的位置上用"[]"作为占位符。在此命令中，options的缺省设置可以用命令options=optimset('fzero')获得。它有两个域：Display和TolX。其中：

- ◆ options.Display　显示设置，有三个选取值[offliterl {final}]；
- ◆ options.TolX　自变量计算的终止误差，可选取[opsitive scalar]，缺省值为2.2204e-016。若要修改缺省值，可用options.TolX=0.001或用options=optimset('TolX',0.001)来改动；
- ◆ P1和P2　是向函数fun传递的附加参数。它的具体取名和函数fun中一致；
- ◆ x　输出参数，为所求的零点自变量值；
- ◆ fval　输出参数，为函数fun在 x 处的值；
- ◆ exitflag　描述函数fnn的退出情况，表明程序的终止条件。若exitflag>0，则表示找到函数零点后退出；若exitflag<0，则表示没有找到零点或在搜索过程中遇到了无穷大的函数值。

【例 6-12】试用 fzero 命令求解函数 $f(x)=x^4-4x-5$ 的零点。

在 MATLAB 内，建立 M 文件 C6L16。

（1）建立函数 $f(x)$ 的 M 文件。

```
function y=fun1(x)                %例 6-12 用 M 函数文件
y=x.^4-4*x-5;
```

（2）建立水平横轴的 M 文件。

```
function y=fun2(x)                %例 6-12 用 M 函数文件（绘出水平轴的函数）
y=0;
```

（3）用作图法估计函数零点位置。

```
fplot('fun1',[-5,5])
hold on
fplot('fun2',[-5,5],'r')
```

程序的运行结果如图 6-1 所示。

- 用 zoom 和 ginput 命令获得零点的初始近似值

在 MATLAB 的绘图窗口中，可实现 zoom 和 ginput 命令进行图形和鼠标的交互操作，或直接在程序中输入下列命令，便可得到如图 6-2 所示局部放大图及鼠标操作线。

```
zoom on                     %局部放大命令
[tt,yy]=ginput(2)           %用鼠标获取 2 个零点猜测值
zoom off                    %恢复原来图形大小
```

显示所得零点初始猜测值，结果为：

```
tt =
    -1.0138
     1.9124
yy =
    0.5848
    0.5848
```

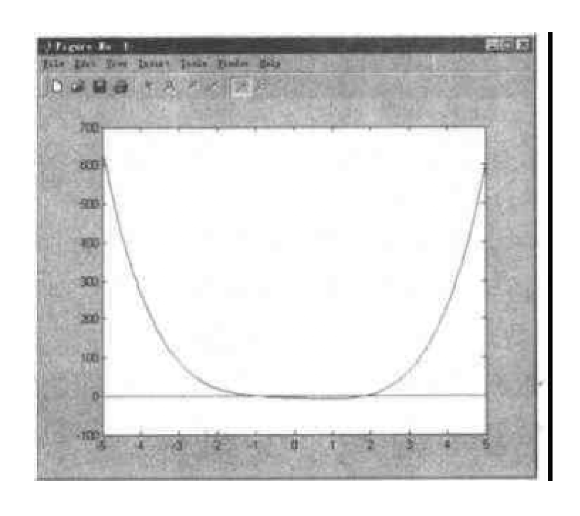


图 6-1 函数零点估计图

● 用函数 fzero 命令求函数的精确零点

```
[x,fval,exitflag]=fzero('fun1',tt(1),[])      %靠近 tt(1)点处的精确零点
[x,fval,exitflag]=fzero('fun1',tt(2),[])      %靠近 tt(2)点处的精确零点
```

结果为：

```
    Zero found in the interval: [-0.98515, -1.0138].
x =
    -1
fval =
       0
exitflag =
          1
Zero found in the interval: [1.8584, 1.9124].
x =
    1.8812
fval =
       -6.2172e-015
exitflag =
          1
```

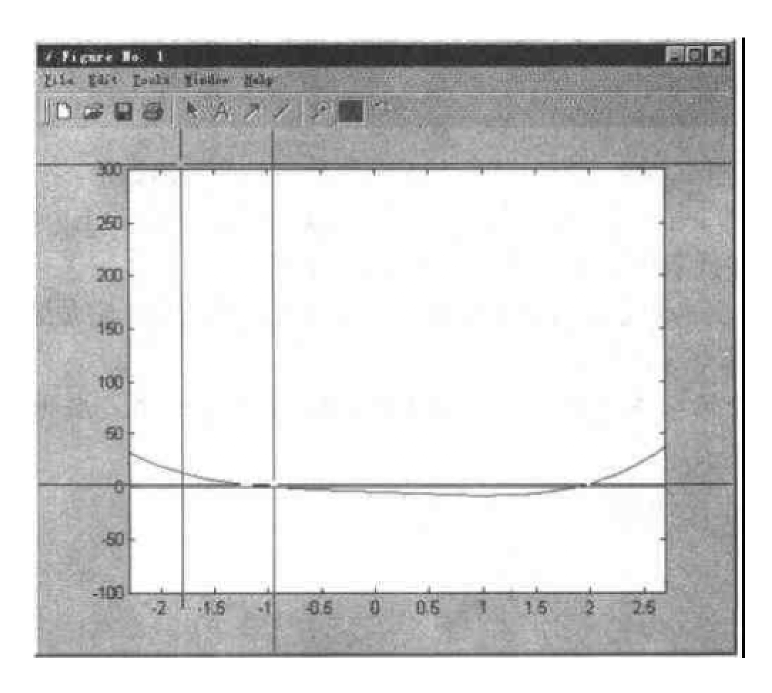


图 6-2 局部放大和利用鼠标取值图

2. 多元函数的零点解非线性方程组

对于采用多元函数的零点解非线性方程组，MATLAB 提供有 fsolve 命令来完成这类工作。一般来说，多元函数的零点问题更难解决。一旦知道零点大致位置，就可用数值方法搜索到精确零点。同一元函数用函数曲线与横轴的交点来获得零点的大致位置一样，二

元函数的零等位线有助于问题的解决。但更高维的初始零点则很难通过图形获得。

有了初始零点后，即可借助函数 fsolve 命令来求零点的精确解。其调用格式为：

- *x*=fsolve(fun,*x*0)　解非线性方程组最简单的调用格式。该式中除两个输入参数外，其余输入输出参数都可以缺省；
- [*x*,fval,exitflag,output,jacob]=fsolve(fun,*x*0,options,*P*1,*P*2...)　解非线性方程组最完整的调用格式。

参数中：

- *x*0 是表示零点数是猜测值的向量。
- options 是 fsolve 的优化迭代所采用参数的结构数组。它的缺省值可用 options=optimset('fsolve')获得。它含有 5 个域：Display、MaxFunEvals、MaxIter、TolFun 和 TolX。其关键参数为 options.Display。
  - ◆ options.Display　显示设置，有 3 个选取值[off|iter|{final}]；
  - ◆ options.MaxFunEvals　允许函数计算的最多次数，可选取[positive integer]，缺省值是自变量的 100 倍；
  - ◆ options.MaxIter　允许的最多迭代次数，可选取[positive integer]，缺省值为 400；
  - ◆ options.TolFun　函数计算的终止误差，可选取[positive scalar]，缺省值为 1.0000e-006；
  - ◆ options.TolX　自变量计算的终止误差，可选取[positive scalar]，缺省值为 1.0000e-006。
- *P*1 和 *P*2　向函数 fun 传递的参数。
- *x* 和 fval　输出参数，分别是所求零点的自变量值和函数值。
- exitflag　若 exitflag>0，则表示找到零点后退出；若 exitflag<0，则表示没有找到零点或在搜索过程中遇到了无穷大的函数值。
- output　输出本命令所用的计算方法、迭代次数等信息；它是结构数组。
- jacob　函数在 *x* 处的 jacobian。

注意：上两例中输入参数 fun 有三种编写方式：字符串方式、内联函数方式和 M 函数文件方式。

【例 6-13】试求方程组$\begin{cases}\sin x+y=0\\ x+6y=0\end{cases}$的解。

（1）观察量函数 0 等位线的交点情况。

```
x=-1:0.5:1;
y=x;
[X,Y]=meshgrid(x,y);        %产生 x-y 平面上的网点坐标
fun1=sin(X)+Y;              %函数 fun1
fun2=X+6*Y;                 %函数 fun2
v=[-0.2,0,0.2];             %指定三个等位线，是为了更可靠地判断 0 等位线的位置
contour(X,Y,fun1,v)         %绘出 fun1 的三条等位线
hold on
```

```
contour(X,Y,fun2,v)      %绘出 fun2 的三条等位线
hold off
```

（2）从图 6-3 中用鼠标获取零点的初始近似值。

```
[x0,y0]=ginput(1);       %用鼠标在图形上取一个点的坐标（三线组中间那条线）
disp([x0,y0])            %显示初始零点猜测值
 0.0000    -0.0058
```

有了初始零点后，便可利用 fsolve 函数命令求精确解。

```
fun='[sin(x(1))+x(2),x(1)+6*x(2)]';     %用字符串表达式形式命令。注意自变量必须写
                                          成 x(1)和 x(2)
XXYY=fsolve(fun,[x0(1),y0(1)])          %解此非线性方程组
disp(XXYY)
XXYY =
          1.0e-016 *
          0.1691    -0.0347
          1.0e-016 *
        0.1691    -0.0347
```

检验

```
FF1=sin(XXYY(1))+XXYY(2)     %检验方程 1
FF2=XXYY(1)+6*XXYY(2)        %检验方程 2
FF1 =
     1.3444e-017
FF2 =
     -3.9031e-018
```

上面的 FF1 和 FF2 表明，所求的零点相当准确。

说明：在使用函数 fsolve 命令时，fun 也可是用内联函数的形式，此时

```
fun=inline('[sin(x(1))+x(2),x(1)+6*x(2)]','x');     %'x'项必须有
XXYY=fsolve(fun,[x0(1),y0(1)])
```

fun 函数也可用 M 函数文件的形式，此时应先用 fun.m 表示被解函数（并在搜索路径上）：

```
function yy=fun(x)
yy(1)=sin(x(1))+x(2);
yy(2)=x(1)+6*x(2);
XXYY=fsolve('fun',[x0(1),y0(1)])
```

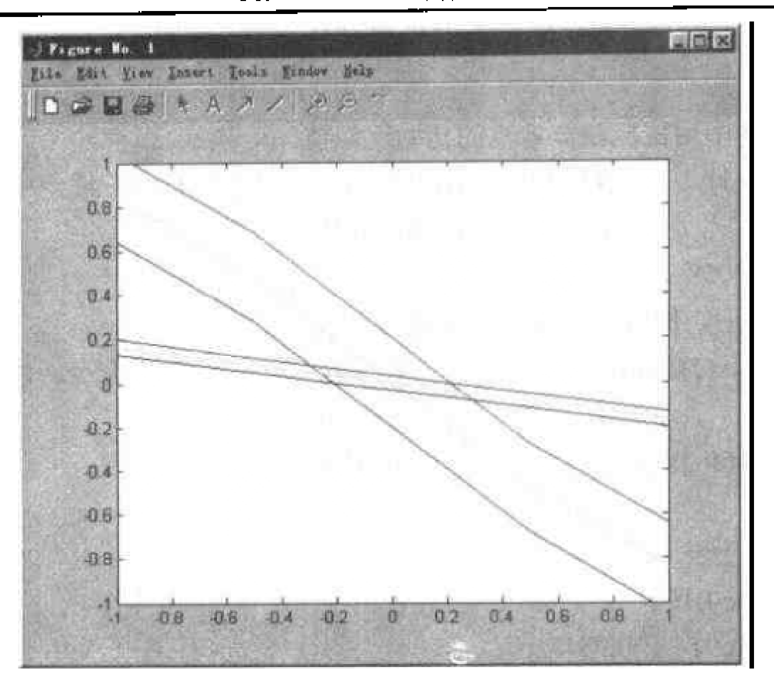


图 6-3　两个二元函数 0 等位线的交点图

6.1.9　符号方程及方程组的求解

在工程教学或工程实践中，会遇到一些带有符号的方程，或某些问题需求它的符号解，这类问题就是符号方程和方程组的求解问题。一般在实际中遇到的代数方程组包括线性方程组、非线性方程组和超越方程组。在 MATLAB 中，用于求解代数方程组的命令有 linsolve 命令和 solve 命令。

1. 线性方程组的符号解

矩阵计算是求解线性方程组最简单有效的方法。从 MATLAB 5.x 版本起，不管数据对象是数值还是符号，实现矩阵运算的指令几乎相同。因而，求线性方程组符号解时，可以套用数值解的命令的编写方法进行。而在 MATLAB 中，函数命令 linsolve 专门用来求解线性方程组，其使用等同于数值计算方法。对方程 *A*X=B*，函数命令 linsolve 的调用格式为：

***X* = linsolve(*A*,*B*)**

等同于 *X* = sym(*A*)\sym(*B*)

矩阵 *A* 必须至少是行满秩的。当 *A* 的列数大于行数时，将给出解不惟一的警告提示。使用该格式得到的方程组特解 *X*。

方程组的通解 *XX* 为：*XX=X+'k'**null(*A*)，其中 *k* 是任意常数；null 命令将求出 *A* 的“零空间”的基。

【例 6-14】求给定线性方程组的解。

```
A=sym('[1 2/3 1/2;3 5 1;1 2 1]')      %输入矩阵 A
A =
    [   1,  2/3,  1/2]
    [   3,    5,    1]
```

```
    [  1,   2,   1]
b=sym('[1 2;1/3 3;1 1/7]');
X=linsolve(A,b)                     %计算线性方程的解
X =
    [  23/39,  283/91]
    [  -8/13, -102/91]
    [  64/39,  -66/91]
```

【例 6-15】求欠定方程组。

```
A=sym('[1 2/3 1/2 1;3 5 1 1;1 2 1 1]')   %输入矩阵 A
A =
    [  1, 2/3, 1/2,   1]
    [  3,   5,   1,   1]
    [  1,   2,   1,   1]
b=sym('[1;1/3;1]');
X=linsolve(A,b)                      %特解
X =
    [-1/3]
    [   0]
    [   0]
    [ 4/3]
XX=X+'k'*null(A)                     %通解
XX =
     [ -1/3-3/2*k]
     [          k]
     [     -8/3*k]
     [ 4/3+13/6*k]
```

说明：欠定方程组的解不惟一，而 MATLAB 总是给出一个特解，并且解向量中含非零元素最少。

2. 一般代数方程组的符号解

slove 命令可以解一般代数方程，包括线性方程、非线性方程和超越方程。当方程不存在符号解，且又无其他自由参数时，函数 solve 将给出数值解。其命令调用格式为：

- solve('eqn1','eqn2',...,'eqn*N*')　对 *N* 个方程的默认变量求解；
- solve('eqn1','eqn2',...,'eqn*N*','var1,var2,...,var*N*')　对 *N* 个方程的 var1,var2,...,var*N* 变量求解，但该式要注意变量的英文字母顺序，并且在变量前不可有空格；
- *S*=solve('eqn1','eqn2',...,'eqn*N*','var1','var2',...,'var*N*')　对 *N* 个方程的'var2',...,'var*N* 变量求解；
- [*x*1,*x*2,…,*xn*]=solve('eqn1','eqn2',...,'eqn*N*','var1','var2',...,'var*N*')　对变量 var1,var2,..., var*N* 求解，并且求解的结果分别赋给 *x*1,*x*2,…,*xn*。此式中，MATLAB 是按照变量 var1,var2,...var*N* 在英文字母中的顺序赋值给 *x*1,*x*2,…,*xn* 的。

提示：'eqn1','eqn2',...,'eqnN' 是字符串表达的方程，或是字符串表达式；'var1','var2',...,'varN'是字符串表达的求解变量名。

第三个命令中，S 是一个结构数组，如果要显示结果，必须使用 S.var1，S.var2，…，S.var$N$ 的引用形式。

在得不到"封闭性解析解"时，将给出数值解。

【例 6-16】求非线性方程组 $\begin{cases} x^2+\sqrt{2}x+2=0 \\ x+3z=4 \\ yz=-1 \end{cases}$ 的解。

```
[x,y,z]=solve('x^2+sqrt(2)*x+2=0','x+3*z=4','y*z=-1','x','y','z')
x =
    [ (-1/2+1/2*i*3^(1/2))*2^(1/2)]
    [ (-1/2-1/2*i*3^(1/2))*2^(1/2)]
y =
  [ -51/73+3/73*i*3^(1/2)-27/146*(-1/2+1/2*i*3^(1/2))*2^(1/2)-3/146*2^(1/2)]
  [ -51/73-3/73*i*3^(1/2)-27/146*(-1/2-1/2*i*3^(1/2))*2^(1/2)-3/146*2^(1/2)]
z =
  [ -1/3*(-1/2+1/2*i*3^(1/2))*2^(1/2)+4/3]
  [ -1/3*(-1/2-1/2*i*3^(1/2))*2^(1/2)+4/3]
S=solve('x^2+sqrt(2)*x+2=0','x+3*z=4','y*z=-1','x','y','z')
S =
    x: [2x1 sym]
    y: [2x1 sym]
    z: [2x1 sym]
disp('S.x'),disp(S.x)          %显示结构数组 S 中 x 的内容
S.x
[ (-1/2+1/2*i*3^(1/2))*2^(1/2)]
[ (-1/2-1/2*i*3^(1/2))*2^(1/2)]
disp('S.y'),disp(S.y)          %显示结构数组 S 中 y 的内容
S.y
[ -51/73+3/73*i*3^(1/2)-27/146*(-1/2+1/2*i*3^(1/2))*2^(1/2)-3/146*2^(1/2)]
[ -51/73-3/73*i*3^(1/2)-27/146*(-1/2-1/2*i*3^(1/2))*2^(1/2)-3/146*2^(1/2)]
disp('S.z'),disp(S.z)          %显示结构数组 S 中 z 的内容
S.z
[ -1/3*(-1/2+1/2*i*3^(1/2))*2^(1/2)+4/3]
[ -1/3*(-1/2-1/2*i*3^(1/2))*2^(1/2)+4/3]
```

【例 6-17】解超越方程组 $\begin{cases} x^x=2 \\ x/y=3 \end{cases}$ 的解。

```
[x,y]=solve('x^x=2','x/y=3')
x =
    log(2)/lambertw(log(2))
y =
    1/3*log(2)/lambertw(log(2))
```

在上述符号解中的 lambertw(ω)代表 ω 函数 $\omega(x)e^{\omega(x)} = x$ 的解。有关详细内容，请用 mhelp 命令访问 MAPLE 库的 lambertw 条目。

6.1.10　矩阵的特征值和特征向量

特征值和特征向量是线性代数中非常重要的概念，在实际的工程应用和求解数学问题中占有非常重要的地位。比如在工程中的振动问题、稳定问题等，从数学关系上常常归结为矩阵的特征值和特征向量的求解问题。在求解微分方程组以及简化矩阵计算等都要用到特征理论。而在与振动有关的各学科中，特征值和特征向量的问题都有着广泛的应用。

矩阵 A 与向量 x 相乘，即表示矩阵对向量的变换（Transformation）。对于任何一个矩阵来说，总存在一些特殊的向量，在变换的作用下，向量的方向不变，仅是长短发生变化。这种向量就是所谓的特征向量（Eigenvector）。它满足方程：$Ax = \lambda x$。

其中，A 是 $n \times n$ 的方阵，λ 称为特征值（Eigenvalue），是个标量。x 是对应 λ 的一个长度为 n 的列向量。该方程就称为特征方程（Eigenvalue Equation）。

上式的广义特征方程是：$Ax = \lambda Bx$。

此式中，A 和 B 都是 $n \times n$ 的矩阵，λ 是标量．当 B=1 时，就退化为 $Ax = \lambda x$。而广义特征值问题就是方程 $Ax = \lambda Bx$ 的非平凡解问题。

一些有缺陷的矩阵不能对角化，不能进行特征值分解。由 A 的特征值构成的对角矩阵，以及由对应的特征向量构成的矩阵 V 的各列，必须满足：$AX=VD$。

如果 V 是非奇异的，则这就是矩阵 A 的特征值分解

1. 矩阵的数值特征值和数值特征向量

在数学教科书上，最常见的求解特征方程 $Ax = \lambda x$ 的方法是：先根据 $|A - \lambda x_i|$=0 求特征值 λx_i（i=1,2,…，n），然后再由 $Ax = \lambda x$ 求对应的特征向量 x_i。而在 MATLAB 中，其计算特征值和特征向量的算法取自 EISPACK 程序库，相应的计算命令较为简单，调用格式如下。

- D = eig(A)　仅计算 A 的特征值组成的列向量；
- [V,D] = eig(A)　生成由矩阵 A 的特征值、矩阵 D 和特征向量构成的矩阵 V，使 $A*V=V*D$；
- [V,D] = eig(A,'nobalance')　计算时不采用预先平衡。通常，预先平衡增加了特征值和特征向量的计算精度，然而，如果一个矩阵包含有与截断误差数量级相差不远的元素时，平衡过程有可能将它们放大，从而导致错误的特征值。该指令可使精度增加；
- D = eig(A,B)　如果 A 和 B 是方阵，生成广义特征值 D；
- [V,D] = eig(A,B)　计算广义特征值矩阵 D 和广义特征向量矩阵 V，使得 $A*V$ =

*B**V*D*。

【例 6-18】计算矩阵 *A* 的特征值和特征向量。

```
A=[-1 1 0;-4 3 0;1 0 2];
B=[1 2 3;1 3 9;4 0 2];
D=eig(A)        %计算矩阵A的特征值
D =
     2
     1
     1
[V,D]=eig(A)
V =
          0        0.4082       -0.4082
          0        0.8165       -0.8165
     1.0000       -0.4082        0.4082
D =
     2     0     0
     0     1     0
     0     0     1
[V,D]=eig(A,B)
V =
     0.5940 - 0.0854i   -0.0356 + 0.5465i   -0.5983 - 0.0466i
    -0.2068 + 0.3408i   -0.0538 + 0.8255i    0.2285 + 0.3267i
    -0.4748 - 0.5054i    0.0081 - 0.1251i    0.4410 - 0.5352i
D =
     0.2524 - 0.5292i                  0                  0
                    0   0.1531 - 0.0000i                  0
                    0                  0   0.2524 + 0.5292i
A*V
ans =
    -0.8008 + 0.4262i   -0.0182 + 0.2790i    0.8269 + 0.3733i
    -2.9965 + 1.3640i   -0.0189 + 0.2905i    3.0789 + 1.1663i
    -0.3556 - 1.0962i   -0.0193 + 0.2964i    0.2836 - 1.1170i
B*V*D
ans =
    -0.8008 + 0.4262i   -0.0182 + 0.2790i    0.8269 + 0.3733i
    -2.9965 + 1.3640i   -0.0189 + 0.2905i    3.0789 + 1.1663i
    -0.3556 - 1.0962i   -0.0193 + 0.2964i    0.2836 - 1.1170i
```

2. 符号特征值和特征向量

在 MATLAB 中，也是使用函数 eig 计算方阵 *A* 的符号特征值和特征向量。其调用格

式为：

Lambda=eig(*A*)

[*V*,*D*]=eig(*A*)

计算任意精度的矩阵特征值和特征向量的调用格式为：

Lambda = eig(vpa(*A*))

[*V*,*D*] = eig(vpa(*A*))

上式中，各参数的具体含义与数值特征值和特征向量相同，在此不再赘述。

【例 6-19】计算符号特征值和特征向量。

```
A=[8/9 1/2 1/3;1/2 1/3 1/4;1/3 1/4 1/5];
A=sym(A)      %转化为符号矩阵
A =
    [ 8/9, 1/2, 1/3]
    [ 1/2, 1/3, 1/4]
    [ 1/3, 1/4, 1/5]
    [V,D]=eig(A)
V =
    [   28/153+2/153*12589^(1/2),      28/153-2/153*12589^(1/2),          1]
    [                          1,                             1,         -4]
    [ 292/255-1/255*12589^(1/2),      292/255+1/255*12589^(1/2),       10/3]
D =
    [ 32/45+1/180*12589^(1/2),                          0,           0]
    [                       0,    32/45-1/180*12589^(1/2),           0]
    [                       0,                          0,           0]
eig(vpa(A))
ans =
     [             -.3179627354760836950300 7e-31]
     [  .87773816609932252988056841486003e-1]
     [       1.3344484056122899692341653807363]
     [V,D] = eig(vpa(A))
V =
    [ .18860838403857944292978827467058,  .80318501188318836927227766194002,
-.56508469644519509785929796275283]
    [
.75443353615431777171915309868189,  .48687247435830155621160586316434,  .4402104
4199100581223229247499660]
    [  .62869461346193147643262758223468,  .3432914656650053566722437372153
1,  .69777793932276550403654035882202]
D =
    [-.33889316547608368542146e-31,                   0,                  0]
```

```
[         0,  1.3344484056122899692341653807363,                    0]
[               0,               0,  .87773816609932252988056841485995e-1]
```

6.1.11 矩阵的对角化和其他矩阵函数

矩阵运算经过矩阵对角化后变得十分简单，特别对于维数较多的矩阵更是如此。矩阵对角化在实际的工程教学和实践中有着非常广泛的用途。几乎所有的对角化都基于特征值和特征向量的求解，特征值和特征向量的求解目的也是为了对角化。

1. 矩阵的 PAP 对角化

由线性代数可知，对于任意可对角化的矩阵 A，都存在一个可逆矩阵 P，使得 $P^{-1}AP$ 为对角阵，并且对角阵的对角线元素为矩阵 A 的特征值。在 MATLAB 中，求出矩阵特征值和特征向量 D 和 V 后，即可满足上述条件（$P=V$）。

【例 6-20】矩阵的 PAP 对角化。

```
A=[3 2 -1;-2 -2 2;3 6 -1];
[P,D]=eig(A)
P =
    0.8890    0.2673   -0.0531
   -0.2540   -0.5345    0.4677
    0.3810    0.8018    0.8823
D =
    2.0000         0         0
         0   -4.0000         0
         0         0    2.0000
inv(P)*A*P
ans =
    2.0000    0.0000   -0.0000
   -0.0000   -4.0000    0.0000
   -0.0000   -0.0000    2.0000
```

2. 实对称矩阵的 QRQ 对角化

实对称矩阵 A 都可对角化，并且都存在正交矩阵 Q，使得 $Q^{-1}AQ$（即 $Q'AQ$）为对角阵。对角阵的对角线元素均为矩阵 A 的特征值。由线性代数的知识可知，对于实对称矩阵 A，特征值分解函数 eig(A)返回的特征向量矩阵就是正交矩阵。

【例 6-21】实矩阵的 QRQ 对角化。

```
A=[2 2 -2;2 5 -4;-2 -4 5];
[Q,D]=eig(A)
Q =
    0.8944    0.3333   -0.2981
   -0.4472    0.6667   -0.5963
```

```
        0      -0.6667      -0.7454
D =
   1.0000            0            0
        0      10.0000            0
        0            0       1.0000
Q'*A*Q
ans =
   1.0000            0            0
        0      10.0000       0.0000
  -0.0000      -0.0000       1.0000
inv(Q)*A*Q
ans =
   1.0000      -0.0000       0.0000
  -0.0000      10.0000      -0.0000
   0.0000      -0.0000       1.0000
```

3. 约旦（Jordan）标准型矩阵

当用相似变换对角化矩阵时，就会产生约旦标准型。即给定矩阵 *A*，寻找非奇异矩阵 *V*，使 inv(*V*)**A***V*（更简洁地说，是使 *J*=*V**A***V*）尽可能地接近对角阵。约旦标准型是特征值矩阵，它的变换矩阵的每一列就是特征向量。对于有重特征值的非对称矩阵，不能进行对角化。约旦标准型对角线上的元素是特征值，但一些对角线以上的元素是 1，而不是 0。

在 MATLAB 中，计算约旦标准型的函数是 jordan，其调用特式为：

***J*=jordan(*A*)**

[*V*,*J*] = jordan(*A*)

式中：*J* 是约旦标准型，*V* 是相似变换矩阵，使得 *V**A***V* = *J*。*V* 的列是 *A* 的一般化特征向量。

【例 6-22】求矩阵的约旦标准型。

```
A=[1 1 1;0 2 1;1 3 1];
[V,J]=jordan(A)
V =
   0.7500    0.0947    0.1553
  -0.2500    0.0947    0.1553
   0.2500    0.1479   -0.3979
J =
   1.0000         0         0
        0    3.5616         0
        0         0   -0.5616
d=eig(A)
```

```
d =
        1.0000
       -0.5616
```

提示：如果是求矩阵 A 的符号约旦标准型，则在输入矩阵 A 时，将 A 以符号矩阵形式输入或把 A 转化为符号矩阵（A=sym(A)）。eig 命令不能用于亏损矩阵。而用 jordan 可求出准确的特征值。

4. 其他矩阵函数

在 MATLAB 中，提供了一些用于矩阵分析和计算的函数，如表 6-2 所示。

表 6-2 常用的矩阵函数

函数名称	功能和含义
cdf2rdf(V, D)	将复数对角形式转化成实数块对角形式
funm(A,'function')	计算矩阵的函数值
eig(A)	对矩阵 A 作特征值分解
hess(A)	矩阵 A 的 hessenberg 形式
expm(A)	求矩阵 A 的指数
null(A)	由奇异值分解得出矩阵 A 的零空间的标准正交基
orth(A)	矩阵 A 行向量的标准正交基
pinv(A)	求矩阵 A 的广义逆
sqrtm(A)	矩阵 A 的平方根
cond(A)	求矩阵 A 的条件数
rref(A)	将矩阵 A 转换为逐行递减的阶梯阵
rsf2csf(V,D)	将实数块对角形式转化为复数对角形式
det(A)	求方阵 A 的行列式值
subspace	计算由 A、B 张成的子空间的夹角
rank(A)	求矩阵 A 的秩
condeig(A)	求对应于矩阵 A 的特征值的条件数
norm(A)	求向量或矩阵 A 的范数

表 6-2 中函数命令的详细使用方法请参见 MATLAB 的帮助文件。

6.2 多项式运算

在实际的工程应用中，都要大量用到多项式，如对实验数据的插值和曲线拟合；在矩阵分析时，也要用到多项式的概念。多项式函数是形式最简单的函数，也是最容易计算的函数，它可以表达绝大多数复杂的函数。本节将介绍 MATLAB 中多项式的表示和各种运算。MATLAB 中的多项式函数位于目录\toolbox\matlab\polyfun 下，它是 MATLAB 提供的基本运算功能之一。

6.2.1 多项式的表示和创建

在 MATLAB 中，任意的多项式都是用一个行向量表示，将多项式的系数按降幂次序

存放在行向量中。多项式 $p(x)=a_0x^n+a_1x^{n-1}+\cdots+a_{n-1}x+a_n$ 的系数行向量为：$P=[a_0\ a_1\ \ldots\ a_n]$，要注意幂的次序必须是从高次到低次。

1. 直接创建多项式

在 MATLAB 中，为了将整个多项式输入 MATLAB，可以采用类似向量的创建方法，直接输入多项式的系数行向量。例如，多项式 x^4+5x^3-8x+3 可以用系数行向量[1 5 0 –8 3]表示。

提示：多项式中缺少的幂次一定要用“0”补齐。

2. 通过有根函数 poly 创建多项式

函数 poly 是由给定的根创建多项式。其调用格式为：

***P*=poly(*A*)**

根据 ***A*** 的不同形式，它有两种返回情况：

- 若 ***A*** 为 $n\times n$ 的矩阵，则返回值 *P* 将是一个含有 *n*+1 个元素的行向量，也就是该矩阵特征多项式的系数；
- 若 ***A*** 为向量，则返回值将是多项式的系数行向量，该多项式的根为 ***A***。

【例 6-23】用函数 poly 建立多项式。

```
A=[6 -8 6; 1 0 0; 0 2 0];
P=poly(A)                  %矩阵 A 的特征多项式的系数
P =
    1.0000    -6.0000    8.0000   -12.0000
roots(P)                   %求多项式的根
ans =
     4.8623
     0.5689 + 1.4644i
     0.5689 - 1.4644i
D=eig(A)                   %求矩阵 A 的特征值
D =
    4.8623
    0.5689 + 1.4644i
    0.5689 - 1.4644i
```

3. 多项式的符号表示

在 MATLAB 的符号工具箱中，对多项式还可以进行符号表示和运算。若矩阵 *A* 是符号矩阵，其调用格式为：

- poly(*A*)　返回符号矩阵 *A* 的特征多项式，其结果是用符号“*x*”或“*t*”表示的多项式变量；
- poly(*A*,*v*)　返回 *A* 的特征多项式，结果是用变量“*v*”来代替变量“*x*”；
- poly2sym(*C*)　将向量 *C* 表示的多项式转化为用符号表示的多项式，其变量用字符“*x*”表示；
- poly2sym(*C*,'*v*') 和 poly2sym(C,sym('*v*'))　将向量 *C* 表示的多项式转化为用符号表

示的多项式，多项式 C 中的变量用符号“v”表示。

【例 6-24】建立符号多项式。

```
syms a b c d v
A=[a b; c d];
poly(A)
ans =
     x^2-x*d-a*x+a*d-b*c
     poly(A,v)
ans =
     v^2-v*d-a*v+a*d-b*c
     C=[1 0 -2 -5];
     poly2sym(C)
ans =
      x^3-2*x-5
      poly2sym(C,'t')
ans =
      t^3-2*t-5
```

6.2.2 多项式的基本运算

MATLAB 提供了多项式运算的一些函数，如表 6-3 所示。

表 6-3 多项式函数

函数名称	功能
conv(a,b)	多项式乘法
[q,r]=deconv(a,b)	除法
poly(A)	用根构造多项式
polyder(a)	对多项式或有理多项式求导
polyfit(x,y,n)	多项式数据拟合
polyval(p,x)	计算 x 点处多项式的值
[r,p,k]=residue(a,b)	部分分式展开式
[a,b]=residue(r,p,k)	部分分式组合
roots(a)	求多项式的根

1. 多项式的乘法 conv

多项式的乘法运算实质就是多项式系数向量的卷积（Convpolytion）运算。长度为 m 的向量 a 和长度为 n 的向量 b 的卷积定义为：

$$c(k)=\sum_{i=1}^{k}a(i)b(k+1-j)$$

此式运算结果（$m+n-1$）为向量 c 的长度。在 MATLAB 中，用函数 conv 来实现其功能。其调用格式为：

c = conv(a, b)

此函数用来求两个多项式的乘积，即两个数组的卷积。

【例 6-25】求多项式 $a = x^3 + 3x^2 + 2x + 1$ 和多项式 $b = 4x^3 + 3x^2 + 9x + 10$ 的乘积。

```
a=[1 3 2 1];
b=[4 3 9 10];
c=conv(a,b)
c =
   4    15    26    47    51    29    10
C=poly2sym(c)          %把多项式的乘积转化为符号多项式的形式
C =
        4*x^6+15*x^5+26*x^4+47*x^3+51*x^2+29*x+10
```

说明：两个以上的多项式相乘，需要重复使用命令 conv。

2. 多项式的除法 deconv

在特殊情况下，一个多项式可除以另一个多项式。多项式的除法运算实质就是多项式系数向量的解卷积（Deconvolytion）运算过程。向量 a 对向量 c 进行解卷积得到“商（Quotient）”向量 q 和“余（Remider）”向量 r，并且满足：

$$c(k) - r(k) = \sum_{i=1}^{k} a(j)q(k+1-i)$$

在 MATLAB 中，用函数 deconv(c,a)来实现这一功能。其调用格式为：

[q,r] = deconv(c,a)

q 和 r 分别是商多项式和余多项式；c 和 a 分别是被除多项式和除多项式，并使得 c = conv(a,q) + r。

【例 6-26】多项式的除法（以例 6-25 中的多项式为例）。

```
a=[1 3 2 1];
c=[4 15 26 47 51 29 10];
[q,r]=deconv(c,a)
q =
     4     3     9    10
r =
     0     0     0     0     0     0     0
Q=poly2sym(q)        %转化为符号多项式的形式，便于阅读
Q =
     4*x^3+3*x^2+9*x+10
```

由此看出，Q 即是例 6-25 中的多项式 b。

3. 多项式的加（减）

在 MATLAB 中，并没有直接进行多项式的加减运算的函数。如果两个多项式向量大小相同，则按数组加减运算规则使用加减运算符进行。如例 6-27。

【例 6-27】多项式的加（减）。

```
a=[1 3 2 1];   b=[4 3 9 10];      %多项式 a 和 b 取自例 6-25
c=a+b                              %进行加法运算
c =
    5     6    11    11
C=poly2sym(c)                      %转化为符号多项式以便于比较
C =
    5*x^3+6*x^2+11*x+11
cj=a-b                             %进行减法运算
cj =
    -3     0    -7    -9
CJ=poly2sym(cj)
CJ =
    -3*x^3-7*x-9
```

4. 多项式的求导 polyder

在 MATLAB 中，函数 polyder 可用来计算多项式的导数，它不仅可计算单个多项式的导数，还可以计算两个多项式相乘和相除的导数。其调用格式为：

- polyder(p)　返回多项式系数向量 p 的导数；
- polyder(a,b)　返回多项式 $a*b$ 的导数，相当于命令 polyder(conv(a,d))；
- [q,d] = polyder(b,a)　返回两个多项式商 b/a 的导数，用 q/d 表示。其中 q 是结果的分子多项式，d 是结果的分母多项式。

【例 6-28】对多项式的求导运算。

```
a=[1 3 2 1];
b=[4 3 9 10];
polyder(a)
ans =
      3     6     2
polyder(a,b)
ans =
    24    75   104   141   102    29
[q,d]=polyder(b,a)
q =
     9    -2   -39   -54   -11
d =
     1     6    13    14    10     4     1
Q=poly2sym(q)
D=poly2sym(d)
Q =
    9*x^4-2*x^3-39*x^2-54*x-11
```

```
D =
   x^6+6*x^5+13*x^4+14*x^3+10*x^2+4*x+1
```

5. 多项式的求值 polyval

在 MATLAB 中，多项式有两种求值方式：按数组运算规则计算多项式的值和按矩阵运算规则计算多项式的值。

函数 polyval 是按数组规则运算，其调用格式为：

***y* = polyval(*p*,*x*)**

该命令是用来计算多项式在 x 处的值。p 是多项式的系数向量，x 是指定自变量的值，可以是标量、向量或矩阵。如果 x 是向量或矩阵，则该函数将对向量或矩阵中的每个元素计算多项式的值，结果是和 x 同维的向量或矩阵。

函数 polyvalm 是按矩阵运算规则计算，其调用格式为：

***y*=polyvalm(*p*,*x*)**

该命令是计算矩阵多项式的值，是用矩阵整体（而不是矩阵元素）作为自变量进行计算。p 是多项式系数向量，相当于用矩阵 x 代替多项式的变量来对矩阵而不是对数组进行计算，x 必须是方阵。

【例 6-29】计算多项式的值。

```
p=[2 3 3 0 6];
x=[1 -1];
y=polyval(p,x)
y =
    14     8
x=[-1 1;1 -1];
y=polyval(p,x)          %按向量运算规则
y =
     8    14
    14     8
ym=polyvalm(p,x)    %按矩阵运算规则
ym =
     16   -10
    -10    16
```

6. 多项式的求根命令 roots

在 MATLAB 中，函数 roots 用于对多项式求根，c 是多项式的系数向量。同时返回多项式的根组成的向量，如果复数根出现，则复数必须成对出现。

【例 6-30】求多项式的根。

```
p=[2 3 3 0 6];
roots(p)
ans =
      -1.2291 + 1.1240i
      -1.2291 - 1.1240i
```

```
        0.4791 + 0.9230i
        0.4791 - 0.9230i
```

7. 多项式的部分分式展开

对于有关多项式的展开问题，MATLAB 提供了 residue 命令来执行部分分式展开和多项式系数之间的转换。其具体的调用形式为：

- [*r*,*p*,*k*] = residue(*b*,*a*) 求多项式之比 $b(s)/a(s)$的部分分式展开，返回值 *r* 是留数，*p* 是极点，*k* 是直向量。*a* 和 *b* 分别是分母和分子多项式的系数向量；
- [*b*,*a*] = residue(*r*,*p*,*k*) 将部分分式展开的形式还原为两个多项式 $b(s)$和 $a(s)$相除的形式。

如果多项式 $a(s)$不含重根，则两个多项式可以写成如下形式：

$$\frac{b(s)}{a(s)}=\frac{r_1}{s-p_1}+\frac{r_2}{s-p_2}+\cdots+\frac{r_n}{s-p_n}+k(s)$$

其中，p_1、p_2、p_3、…、p_n 称为极点，r_1、r_2、r_3、…、r_n 称为留数，$k(s)$是直项。留数和极点满足 n = length(a)-1 = length(r) = length(p)。如果 b 的次数小于 a 的次数，则直项的系数向量的系数为空，否则，它们之间满足 length(k) = length(b)-length(a)+1。

如果 $a(s)$含有 m 个重根 $P(j)=\ldots=P(j+m-1)$，则展开的这 m 项应写为：

$$\frac{r_j}{s-p_j}+\frac{r_{j+1}}{(s-p_j)^2}+\cdots+\frac{r_{j+m-1}}{(s-p_j)^m}$$

【例 6-31】多项式的部分分式展开。

```
a=[2 4 5 8 2];
b=[3 5 8 3];
[r,p,k]=residue(b,a)        %将多项式的除部分分式展开
r =
    0.7921
    0.2683 - 0.3191i
    0.2683 + 0.3191i
    0.1713
p =
    -1.7071
    -0.0000 + 1.4142i
    -0.0000 - 1.4142i
    -0.2929
k =
    [ ]
[bb,aa]=residue(r,p,k)          %将部分分式展开还原
bb =
      1.5000      2.5000      4.0000      1.5000
```

```
aa =
      1.0000    2.0000    2.5000    4.0000    1.0000
```

6.2.3 因式分解和展开

在 MATLAB 中提供了对符号表达式、符号矩阵等进行因式分解、同类项合并和展开等操作。因式分解的主要函数有 factor、collect 和 expand，下面将分别介绍这些函数。

1. factor 函数

函数 factor 的调用格式为：

factor(*s*)

若 *s* 是个多项式、系数为有理数，则给出命令将 *s* 表示成系数与有理数的低阶多项式相乘；若 *s* 不能被分解成有理式，则返回的结果就是 *s* 本身；若 *s* 是矩阵，则对矩阵 *s* 中的每个元素进行介解。此外，factor 命令也可对符号整数进行因式分解。

【例 6-32】factor 函数的用法。

```
syms x
s=x^9-1;
factor(s)      %对 s 进行分解
ans =
      (x-1)*(x^2+x+1)*(x^6+x^3+1)
s1=[s;x^6-1];
factor(s1)    %对矩阵进行分解
ans =
      [        (x-1)*(x^2+x+1)*(x^6+x^3+1)]
      [ (x-1)*(x+1)*(x^2+x+1)*(x^2-x+1)]
factor(sym('12345678901234567890'))          %对符号整数进行分解
ans =
      (2)*(3)^2*(5)*(101)*(3803)*(3607)*(27961)*(3541)
```

2. 同类项合并函数 collect

collect 函数的调用格式为：

- collect(*S*)　将 *S* 表示成关于符号变量的多项式形式，该命令将 *S* 中的 *x*（缺省）的同幂项系数进行合并；
- collect(*S*,*v*)　把 *S* 中 *v* 的同幂项系数进行合并。

【例 6-33】collect 函数的使用。

```
syms x y
S=x^2*y + y*x - x^2 - 2*x;
collect(S)
ans =
      (y-1)*x^2+(y-2)*x
collect(S,y)
ans =
```

```
        (x^2+x)*y-x^2-2*x
f = -1/4*x*exp(-2*x)+3/16*exp(-2*x);
collect(f,exp(-2*x))
ans =
        (-1/4*x+3/16)*exp(-2*x)
```

表达式展开函数 expand 其调用形式为：

expand(s)

用于将表达式展开。

【例 6-34】expand 函数的应用。

```
syms x y
expand((x+1)^3)
ans =
        x^3+3*x^2+3*x+1
expand(sin(x+y))
ans =
        sin(x)*cos(y)+cos(x)*sin(y)
v = [exp(x + y) log(x^2/y)];
expand(v)
ans =
        [ exp(x)*exp(y),       log(x^2/y)]
```

6.2.4 多项式的简化

在工程实践和教学计算中，经常涉及到一些公式的推导和化简问题，而这些问题不论是用手在纸上进行，还是用 MATLAB 计算，都会出现将前面已知的式子代入新式和对初步运算结果进行化简的情况，如果式子较复杂，不论是化简还是代入，都容易出错。在 MATLAB 中，可使用 pretty、simplify、simple 和 horner 命令来处理此类问题。

1. pretty 命令

命令 pretty 是将代数式转化为手写格式，即把 MATLAB 进行的幂次运算、乘方运算以及式子中的"*"和"^"符号等化简为平常手写格式。其调用格式为：

- pretty(*s*) 将代数式 *s* 转化为手写格式，并且在转化过程中不会对 *s* 进行任何的化简或展开；
- pretty(*s*,*n*) 用屏幕宽度 *n* 来代替缺省值 79。

【例 6-35】函数 prretty 命令的使用。

```
syms x y
f=x*(x*(x-y+6)+11)^3+4;
pretty(f)
                          3
          x (x (x - y + 6) + 11)  + 4
```

2. simplify 单一化简命令

simplify 函数是一个强有力的、具有普遍意义的工具。它可以对包含有求和、积分、方根、分数的乘方、三角函数、指数函数、对数函数、Bessel 函数、hypergeometric 函数和 gamma 函数的表达式，经过计算机比较后，化简为一种相对简单的形式。用户并不知道化简结果这种形式经过何种变换后得到。在一般的化简中，simplify 函数命令是一种简便快捷的化简方法。

simplify 函数的调用格式为：

simplify(*s*)

将 *s* 进行化简，如果 *s* 是符号矩阵，则将 *s* 中的每一个元素进行化简。

【例 6-36】simplify 函数的用法。

```
syms x c alpha beta
simplify(sin(x)^2 + cos(x)^2)
ans =
    1
simplify(exp(c*log(sqrt(alpha))))
ans =
    alpha^(1/2*c)
```

3. simple 函数

simple 函数的目的是寻找表达式的最简形式，即含有最少的字符。它综合了 pretty simplify 化简方法的优点，通过将 simplify、collect、factor、horner 和其他简化函数应用于表达式并记录结果的长度，最后从中选择最短的结果。其函数调用格式为：

- simple(*s*) 对符号表达式 *s* 使用不同的化简方法，显示化简的中间过程（包括每种化简方法和相应的化简结果），最后从中选择一个最短的结果作为返回值；如果 *s* 是矩阵，则返回结果是整个矩阵的最短表示形式，但不一定是每个单独元素的最短表示；
- [*R*,How] = simple(*s*) 该命令不显示简化过程，返回最短的结果给输出参数 R，同时返回最短结果的形式所用的简化方法给输出参数 How；
- 在 MATLAB 中，simple 命令有几个转化运算。即：
  - combine(trig) 以三角函数运算性质为主对表达式进行化简；
  - convert(exp) 将表达式尽量转化为由 ex、eix 表示的指数表达式；
  - convert(sincos) 将表达式尽量转化为由 sin(*x*)和 cos(*x*)表示的式子；
  - convert(tan) 将表达式尽量转化为由 tan(*x*)表示的式子。

【例 6-37】函数 simple 的用法。

```
syms x
s=2*cos(x)^2-sin(x)^2;
simple(s)                          %使用没有输出参数的命令格式
%下面是化简的中间过程
simplify:
3*cos(x)^2-1
```

```
radsimp:
2*cos(x)^2-sin(x)^2
combine(trig):
3/2*cos(2*x)+1/2
factor:
2*cos(x)^2-sin(x)^2
expand:
2*cos(x)^2-sin(x)^2
combine:
3/2*cos(2*x)+1/2
convert(exp):
2*(1/2*exp(i*x)+1/2/exp(i*x))^2+1/4*(exp(i*x)-1/exp(i*x))^2
convert(sincos):
2*cos(x)^2-sin(x)^2
convert(tan):
2*(1-tan(1/2*x)^2)^2/(1+tan(1/2*x)^2)^2-4*tan(1/2*x)^2/(1+tan(1/2*x)^2)^2
collect(x):
2*cos(x)^2-sin(x)^2
%下面是最终结果
ans =
     3*cos(x)^2-1
[R How]=simple(s)            %使用有输出参数的命令格式
R =
   3*cos(x)^2-1
How =
      simplify
```

4. horner 函数

horner 函数是一种很特别的表达式化简方法，即重迭法。它的化简方法是将表达式尽量化为用括号括起的连乘积形式。在 MATLAB 中，其命令调用格式为：

horner(*p*)

将符号表达式转化为嵌套形式的表达式。

【例 6-38】函数 horner 的用法。

```
syms x
p=x^3-6*x^2+11*x-6;
horner(p)                    %用函数 horner 命令进行化简
x*(x*(x-6)+11)-6
```

6.2.5 多项式的提取和替换

在 MATLAB 提供了通过符号替换简化的表达式输出形式功能，从而得到一个较为简

单的表达形式。实现该功能的函数是 subexpr 和 subs。这两条命令各有优势，用起来也较为方便，下面分别予以介绍。

1. subexpr（提取）命令

MATLAB 中的 subexpr 命令可以将复杂的式子通过筛选相同因子和整理式子化简表达式。其调用格式为：

[*Y*,SIGMA] = subexpr(*X*,SIGMA)或[*Y*,SIGMA] = subexpr(*X*,'SIGMA')

式中：

- *X* 待整理的表达式或表达式矩阵；
- *SIGMA* 在整理过程中提出各种因子将以矩阵的格式保存在名为 *SIGMA* 的变量中；
- *Y* 经提取各种因子后，整理完毕的表达式或矩阵将被保存在名为 *Y* 的矩阵中。

【例 6-39】subexpr 命令的用法。

```
t = solve('a*x^3+b*x^2+c*x+d = 0');
[r,s] = subexpr(t,'s')     %进行提取
r =
    [                                                              1/6/a*s^(1/3)-
2/3*(3*c*a-b^2)/a/s^(1/3)-1/3*b/a]
    [                                  -1/12/a*s^(1/3)+1/3*(3*c*a-b^2)/a/s^(1/3)-
1/3*b/a+1/2*i*3^(1/2)*(1/6/a*s^(1/3)+2/3*(3*c*a-b^2)/a/s^(1/3))]
    [                          -1/12/a*s^(1/3)+1/3*(3*c*a-b^2)/a/s^(1/3)-1/3*b/a-
1/2*i*3^(1/2)*(1/6/a*s^(1/3)+2/3*(3*c*a-b^2)/a/s^(1/3))]
s =
    36*c*b*a-108*d*a^2-8*b^3+12*3^(1/2)*(4*c^3*a-c^2*b^2-
18*c*b*a*d+27*d^2*a^2+4*d*b^3)^(1/2)*a
```

此时可以用命令 pretty 将符号表达式转化为手写形式。

2. subs（替换）命令

在 MATLAB 中，将一表达式代入另一式中的操作命令是 subs。它的用法较为灵活，并且适用范围较广。其调用格式为：

- subs(*S*) 用当前 MATLAB 工作空间中的变量替换 *S* 中的所有同名符号；
- subs(*S*,new) 用符号 new 替换 *S* 中的所有同名符号；
- subs(*S*,old,new) 用符号 new 替换 *S* 中所有的 old 符号。

其中，*S* 是表达式；old 是表达式 *S* 中将要被替换的旧变量名；new 是将要替换 old 的新变量或表达式。

【例 6-40】函数 subs 的使用。

```
a=980;
C1=3;
y = dsolve('Dy = -a*y')      %解微分方程
y =
    C1*exp(-a*t)
```

```
R1=subs(y)                  %用 MATLAB 窗口中的变量 a=980 和 C1=3 分别替换 y 中
                             的 a 和 C1
R1 =                        %输出结果
    3*exp(-980*t)
syms a b t x y
R2=subs(a+b,a,4)            %用 4 替换 a+b 中的 a
R2 =
    4+b
subs(cos(a)+sin(b),{a,b},{sym('alpha'),2})    %用符号 alpha 和 2 分别替换 cos(a)+sin(b)
                                                中的 a 和 b
ans =                       %结果为
cos(alpha)+sin(2)
subs(exp(a*t),'a',-magic(2))   %用魔鬼方阵中每一元素替换表达式 exp(a*t)中的符号 a
ans =
[    exp(-t), exp(-3*t)]
[ exp(-4*t), exp(-2*t)]
subs(x*y,{x,y},{[0 1;-1 0],[1 -1;-2 1]})    %用单元数组替换 x 和 y 并进行计算
ans =
     0    -1
     2     0
whos                                      %用 whos 命令查看工作空间的变量
  Name        Size          Bytes   Class

  C1          1x1               8   double array
  R1          1x1             150   sym object
  R2          1x1             130   sym object
  A           1x1             126   sym object
  Ans         2x2              32   double array
  B           1x1             126   sym object
  T           1x1             126   sym object
  X           1x1             126   sym object
  Y           1x1             126   sym object
Grand total is 33 elements using 950 bytes
```

由此可以看出，除了 C1 和 ans 是双精度变量外，其余都是符号表达式。

注意：subs 命令在进行多个变量和多个矩阵同时替换时，调用格式完全相同，只是进行替换新老变量时要分别用大括号括起来，即用单元数组进行替换；在 subs 命令中，old 可以取字符串表达式、符号变量和单元数组。

6.3 曲线拟合

当存在大量的实验数据时，面对许多杂乱无章的数字，往往无从下手，更无从发现其内在规律，此时运用 MATLAB 的曲线拟合功能，问题便可得到很好地解决。

6.3.1 多项式拟合

对于给定的一组数据$\{(x_i,y_i),\ i=1,2,\cdots,n\}$，如果要采用多项式模型对数据组进行描述，形成如多项式 $y(x)=f(x,a)=a_1x^n+a_2x^{n-1}+\ldots+a_nx+a_{n+1}$ 的形式，求取参数 a_i，使得量值 $\chi^2(a)$ 的值最小的过程，称为对数据组进行多项式拟合，其中：

$$\chi^2(a)=\sum_{i=1}^{n}\left(\frac{y_i-f(a,x_i)}{\Delta y_i}\right)$$

MATLAB 系统设计 polyfit 函数采用最小二乘法原理对给定的数据组$\{(x_i,y_i),i=1,2,\cdots,n\}$进行多项式的曲线拟合，最后给出拟合的多项式系数。函数的调用格式为：

p=polyfit(*x*,*y*,*n*)

其中 *x*，*y* 为数据的横、纵坐标向量；*n* 是给定的拟合多项式的阶数，返回一个多项式 *p*(*x*)的系数向量 *p*。

- [*a*,*S*]=polyfit(*x*,*y*,*n*)：对于所给数据，计算 *n* 阶拟合多项式的系数 *a*；
- [ye,delta]=polyval(*a*,*x*,*S*)：利用估计量计算数据带 $\bar{y}_i\pm\Delta y_i$。

x，*y*，*n* 含义同上；*a* 为所得多项式的系数；*S* 是一个由范德蒙矩阵（Vandermonde）进行克劳斯（Cholesky）分解的系数矩阵、自由度 df 和余数的 2-范数（normr）组成的结构矩阵，供 polyval 使用；ye 是原数据 y_i 的估计。delta 为误差。

注意：由于范德蒙矩阵（Vandermonde）的条件数随着阶数的提高而迅速变大，为保证计算的精度，拟合多项式的阶数一般不要超过五阶。拟合多项式只是在原始数据范围内有可以保证的精度，超出原始数据范围使用多项式预报，要谨慎进行。

【例 6-41】对给定的数据进行多项式数据拟合。

```
x=1:20;
y=sqrt(x)+sin(x);
p=polyfit(x,y,5)
[a,S]=polyfit(x,y,5)
plot(x,y,'o',x,polyval(p,x),'-')
p =
    0.0000   -0.0009    0.0079    0.0222   -0.2685    2.2368
```

a =

0.0000 -0.0009 0.0079 0.0222 -0.2685 2.2368

S =

R: [6x6 double]

df: 14

normr: 2.7895

绘图结果显示如图 6-4 所示。

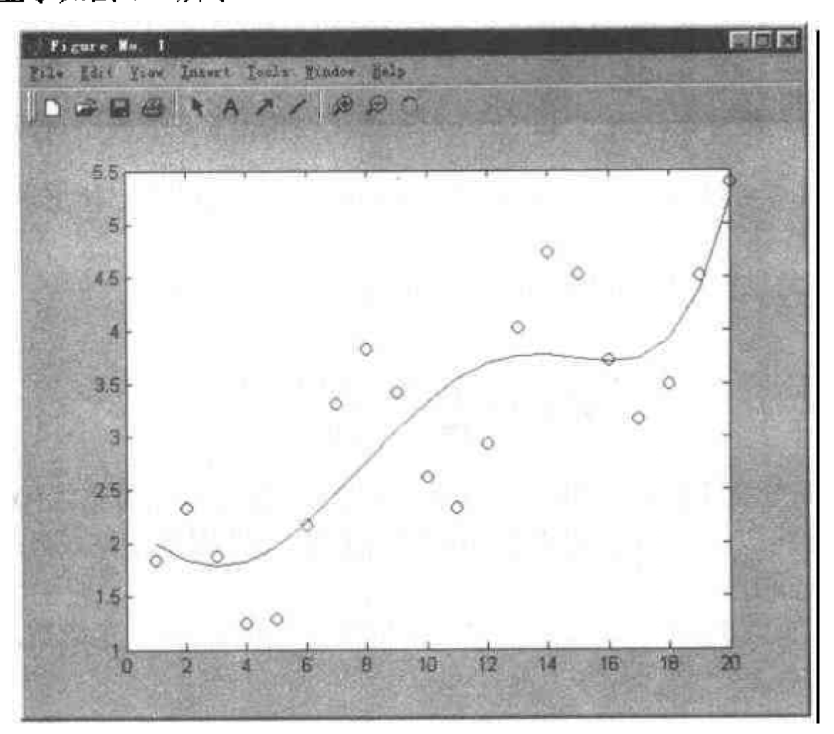


图 6-4 拟合阶数 $n=5$ 时的拟合曲线

其他保持不变，改变阶数 n 为 2 的结果如下，其图形显示如图 6-5 所示。

p =

-0.0000 0.1592 1.4658

a =

-0.0000 0.1592 1.4658

S =

R: [3x3 double]

df: 17

normr: 3.0643

由于数据较少，图 6-4 数据离散较大，由本例可见，当改变拟合阶数时，对拟合的结果影响也较大。

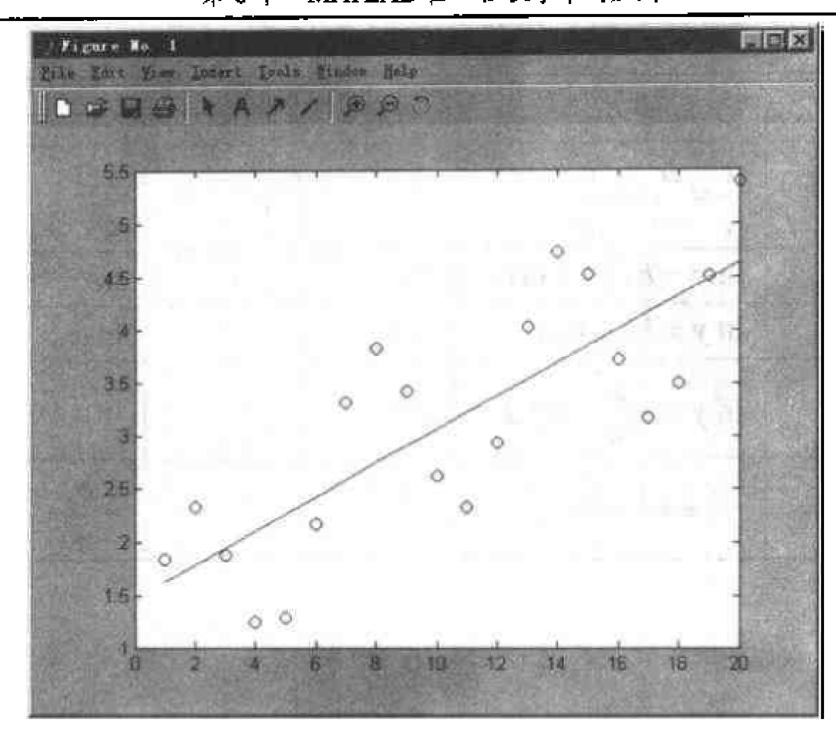


图 6-5 拟合阶数 $n=2$ 时的拟合曲线

6.3.2 非线性最小二乘估计

当模型中的被估计参数与函数值为非线性关系时，对原始数据的拟合就成了非线性最小二乘的问题。由于非线性问题其解的存在性和是否惟一难以确定，如果通过函数变化能使被估计参数以线性形式出现，则应优先使用函数变换，将原问题转换为可以简单有效处理的线性最小二乘问题。一般而言，不要轻易使用非线性最小二乘命令，哪怕是部分参数可以线性化的也应先行处理，尽量减少非线性工作的数量。

非线性最小二乘问题与最小值问题相似，但又有所区别。非线性最小二乘由于建模总是用比较简单的模型函数，所以其导数总是可以求得的。这就有利于构造比较完善的专用算法。$\chi^2(x)$ 函数的最小值点（即最小二乘点）总是处于浅坦、平缓的凹区中。因此不能盲目地追求高精度解。通常，解的精度是 2~3 位有效数字。

线性最小二乘的处理方法是简单的矩阵除法，比较快捷和可靠。因此，即便 MATLAB 中有许多通用的非线性参数估计命令，本方法也值得优先考虑，尤其对于被估计参数较多的情况更该如此。MATLAB 中常见的可以线性化的模型如表 6-4 所示。

表 6-4 可线性化的非线性模型

模型形式	变换后的形式	变量和参数的变化			
		Y	X	$a1$	$a2$
$y=\dfrac{ax}{1+bx}$	$\dfrac{1}{y}=\dfrac{1}{ax}+\dfrac{b}{a}$	$\dfrac{1}{y}$	$\dfrac{1}{x}$	$\dfrac{1}{a}$	$\dfrac{b}{a}$
$y=\dfrac{a}{x-b}$	$\dfrac{1}{y}=\dfrac{x}{a}-\dfrac{b}{a}$	$\dfrac{1}{y}$	x	$\dfrac{1}{a}$	$-\dfrac{b}{a}$

续表

模型形式	变换后的形式	变量和参数的变化			
		Y	X	$a1$	$a2$
$y=\dfrac{ax}{b^2-x^2}$	$\dfrac{x}{y}=\dfrac{b^2}{a}-\dfrac{x^2}{a}$	$\dfrac{x}{y}$	x^2	$\dfrac{1}{a}$	$\dfrac{b^2}{a}$
$y=ax^b$	$\ln y=b\ln x+\ln a$	$\ln y$	$\ln x$	$\ln a$	b
$y=ae^{bx}$	$\ln y=bx+\ln a$	$\ln y$	x	$\ln a$	b
$y=ax^{-x^2/b^2}$	$\ln y=-\dfrac{x^2}{b^2}+\ln a$	$\ln y$	x^2	$\ln a$	$-\dfrac{1}{b^2}$
$\dfrac{x^2}{a^2}+\dfrac{y^2}{b^2}=1$	$y^2=b^2-\dfrac{b^2}{a^2}x^2$	y^2	x^2	$-\dfrac{b^2}{a^2}$	b^2

6.4 插值和样条

多项式曲线拟合是研究如何找到一条“平滑”的曲线，以便最好地表现带噪声的“测量数据”，并不要求拟合曲线通过这些“测量数据”点。而插值（Interpolarion）是在原始数据点之间按照一定的关系插入新的数据点，以便更准确地分析数据的变化规律。它是在假定所给的“基准数据”完全正确的情况下，研究如何“平滑”地估算出“基准数据”之间其他点的函数值。

最简单的插值方法是先根据基准数据，调用 MATLAB 的绘图命令获得数据的图形表现，然后再估计所需点处的值。实际上，MATLAB 在绘图时，总是用直线将两个相邻点连接起来，构成一条完整的曲线。MATLAB 的插值函数如表 6-5 所示，它们位于目录 \toolbox\matlab\polyfun 中，属于 MATLAB 提供的基本运算功能。

表 6-5 MATLAB 的插值函数

函数名	含义和功能
interp1	一维插值函数
interpft	利用 FFT（快速傅里叶变换）的一维插值函数
interp2	二维插值函数
interp3	三维插值函数
interpn	高维插值函数
griddata	规则化数据和曲面拟合
meshgrid	产生“经纬”矩阵
ndgrid	产生高维“经纬”矩阵
spline	样条插值函数

提示： 关于插值和样条的高级应用命令可参见样条工具箱（Spline Toolbox）。

6.4.1 一维插值

一维插值就是对一维函数 $y=f(x)$的数据进行插值，在 MATLAB 中相应的一维插值函

数为 interp1，其调用格式为：

yi=interp1(x,y,xi,method)

其中，输入参数为原始数据点(x,y)，x 为横坐标向量，y 为纵坐标向量。x 的数据必须按单调方式排列。如果 y 为矩阵，则插值将按照 y 的列向量进行，返回值 yi 将和矩阵 y 的列向量相等。

xi 为指定插值点的横坐标，yi 是在 xi 指定位置计算出的插值结果。如果 xi 的某元素 xi(i)超出 x 的定义范围，那么相应的 yi(i)将取值为 NaN。

输入参数"method"用于指定插值的方法，缺省时默认采用线性插值法。MATLAB 提供四种插值方法以供选择，其各自特点如表 6-6 所示。

表 6-6　**四种插值方法列表**

method	含义	特点和用途
linear	线性插值	仅用作连接图上的数据点，速度较快，有足够精度，最常用，作为缺省设置
cubic	三次多项式插值	速度较慢，精度高，平滑性好，常作为平滑使用
spline	三次样条插值	速度最慢，精度最高，最平滑，常作为平滑使用
nearst	最近邻域插值	速度最快，占用内存最少，精度最低；实时使用，特大数据量处理，需要保持基准数据而又不增加新函数值的特殊场合

【例 6-42】对由正弦函数生成的原始数据点进行一维插值。

```
x=0:10;                    %给出 x 的定义范围
y=sin(x);
xi=0:0.2:15;               %设置插值点的横坐标
yi1=interp1(x,y,xi);       %分别按照四种不同的方法选择插值
yi2=interp1(x,y,xi,'nearst');
yi3=interp1(x,y,xi,'spline');
yi4=interp1(x,y,xi,'cubic');
subplot(2,2,1)             %按照子图的方式分别绘制插值结果和原点数据（用'o'表示）
plot(x,y,'o',xi,yi1)
title('线性插值')
subplot(2,2,2)
plot(x,y,'o',xi,yi2)
title('最近邻域插值')
subplot(2,2,3)
plot(x,y,'o',xi,yi3)
title('三次样条插值')
subplot(2,2,4)
plot(x,y,'o',xi,yi4)
title('三次多项式插值')
```

程序运行结果如图 6-6 所示。

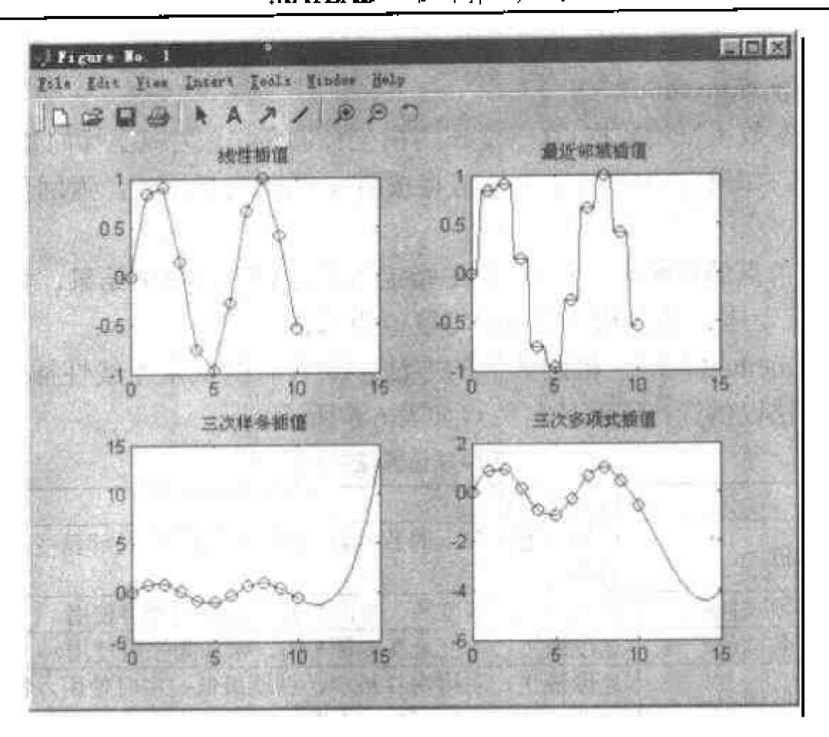


图 6-6　一维插值图形

从图 6-6 中可以看出，对于同样的数据点，不同的插值方法得到的结果并不相同。这是因为插值本身就是一个估计或猜测的过程，应用不同的估计规则便会导致不同的结果。样条插值的结果更平滑，但并不一定更精确。同样由图 6-6 中可以看出，由于程序中所设置的插值点的横坐标有在原始数据点范围之外的点（>10），而只有样条插值可以估计原始数据点外侧的值，虽然它并不一定准确。

6.4.2　二维函数插值

二维函数插值基于同一维函数插值一样的基本思想，它是对两个变量的函数 $z=f(x,y)$ 进行插值。二维插值函数命令为 interp2，其调用的基本格式为：

***zi*=interp1(*x*,*y*,*z*,*xi*,*yi*,method)**

其中，输入参数为原始数据点(x,y,z)；x、y 为两个独立的向量，它们必须是按照单调方式排列的；z 为矩阵，是由 x，y 确定的点上的值。z 和 x、y 的关系为：

$z(i,:)=f(x,y(i))$和 $z(:,i)=f(x(i),y)$

即当 x 变化时，z 的第 i 行与 y 的第 i 个元素 $y(i)$相关，当 y 变化时，z 的第 i 行与 x 的第 i 个元素 $x(i)$相关。如果省略 x 和 y，就相当于假定 x=1:n，y=1:m，此处 n、m 分别是矩阵 z 的行数和列数。

xi 为指定插值点横坐标的数值数组，*yi* 是指定插值点纵坐标的数值数组。

输入参数"method"用于指定插值的方法，同样有四种可供选择的插值方法，与一维插值相同（见表 6-6）。缺省时默认采用"linear"表示双线性插值法（Bilinear Interpolation）。在此就不赘述。

【例 6-43】对由函数 peaks 产生的三维高斯分布进行二维插值。

```
[x,y]=meshgrid(-3:0.3:3);
```

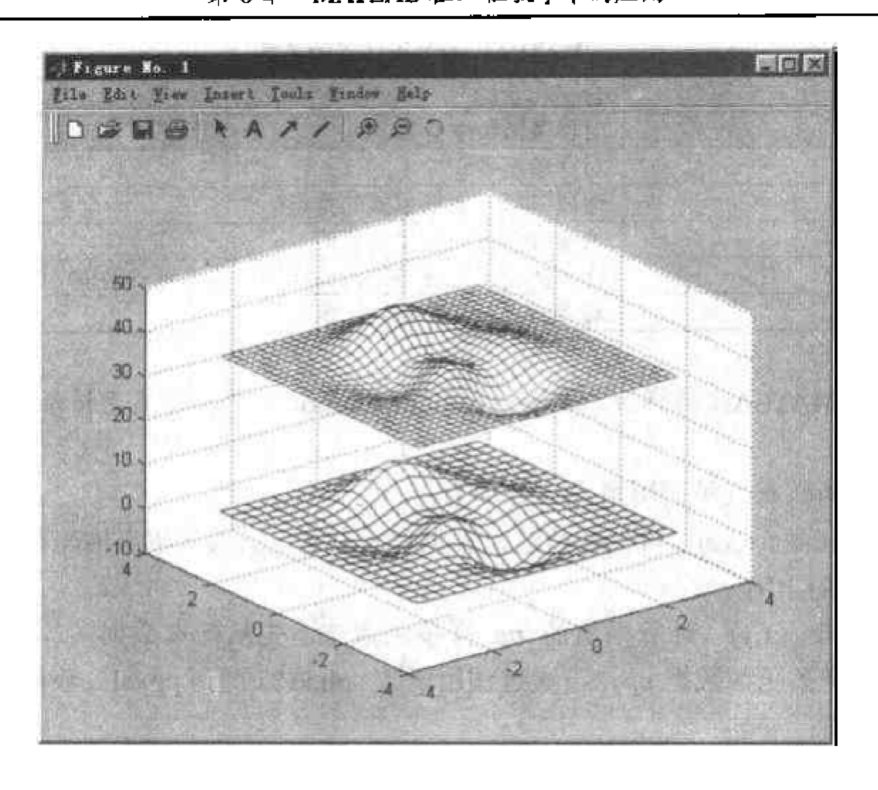


图 6-7 二维插值

```
[x,y]=meshgrid(-3:0.3:3);
z=peaks(x,y);
[xi,yi]=meshgrid(-3:0.2:3);
zi=interp2(x,y,z,xi,yi,'spline');
mesh(x,y,z)
hold on
    mesh(xi,yi,zi+35)
hold off
```

程序的运行结果如图 6-7 所示。

通过图 6-7 可以看出，由于网格划分的大小不同，画出的图形精细程度也不同，因此可以通过插值点划分的情况来绘制出更为精细的图形。

6.4.3 样条函数插值

样条函数产生的基本思想是：设有一组已知的数据点，目标是找出一组拟合多项式，以便用更光滑的曲线来拟合数据点。最常用的方法是用一个三阶多项式来对相邻数据点(在样条术语中称为断点 Breakpoints）之间的各段建模。每个三次多项式的前两个导数与该数据点相一致，这类插值称为三次样条或者简称为样条。同时，为了保证拟合的惟一性，对该三次多项式的一阶和二阶导数加以约束，使其在断点处相等。这样除了在被研究区间端点外，所有内断点处保证样条有连续的一阶和二阶导数。

MATLAB 关于样条的函数名称及其功能如表 6-7 所示。

表 6-7 MATLAB 的关于样条的函数

函数名称	含义和功能
finder	对样条函数求导数
fnint	对样条函数积分
mkpp	逐段多项式数据形式的重组
ppval	由逐段多项式数据求插值
spline	样条插值，或者获得构成样条逐段多项式的值
unmkpp	对逐段多项式数据“反重组”

提示：在 MATLAB 的样条工具箱（Spline Toolbox）中，有关于样条的更多专门函数命令。

样条插值函数有三种调用格式，分别为：

- *yy*=spline(*x*,*y*,*xx*) 直接插值法，根据样本数据(*x*,*y*)求插值点横坐标 *xx* 所对应的三次样条插值；
- *pp*=spline(*x*,*y*) 计算插值的 *pp* 样条函数法，先从样本数据（*x*，*y*）获得逐段多项式样条函数数据 *pp*，再通过调用函数 spline 来得到 ppval，获得横坐标 *xx* 对应的插值；
- *yy*=ppval(*pp*,*xx*) 根据逐段多项式样条函数数据 *pp*，计算 *xx* 对应的函数值 *yy*。

一般情况下使用第一种调用方式。spline 函数依据样本数据（*x*，*y*）及插值点横坐标 *xx*，寻求拟合 *x* 和 *y* 的三次样条内插多项式，然后计算这些多项式，对每个 *xx* 的值，寻找相应的 *yy*。该方式适合于只需要一组内插值的情况。如果需要从相同数据里获得另一组内插值，再次计算三次样条系数便没有意义。此种情况下，便可以采取第二种调用方式，此时，spline 函数根据样本数据（*x*，*y*）返回一个称之为三次样条的 *pp* 形式或分段多项式形式的数组；调用函数 ppval，通过 *pp* 形式参数，计算插值点 *xx* 对应的函数值 *yy*。因此可以说，对于同一组样本数据和相同待求参数 *xx*，直接插值法和计算插值的 *pp* 样条函数法的执行结果相同。

提示：直接插值法在每次插值时必须直接调用样本数据；而一旦 *pp* 样条函数生成，计算插值时，就不再调用样点，而且可以像解析函数那样直接进行微分和积分。

对于相同的 *pp* 形式，指定不同 *xi* 可以得到不同的三次样条插值。当要计算三次样条表示时，必须把 *pp* 形式分解成它的各个部分。在 MATLAB 中通过函数 unmkpp 完成此项功能，其调用格式如下：

[breaks,coefs,pieces,order,dim]=unmkpp(*pp*)

输出参数的个数可以是 1 个到 5 个之间的任何数，函数自动按照式中的次序输出这些内容，输出参数缺省时只返回第一个参数，即断点位置（breaks）。

反之，如果给定 ***pp*** 形式的各个部分，可以用函数重构为完整的 *pp* 形式。

【例 6-44】练习样条函数的用法。

```
x = 0:10;  y = sin(x);          %给出原始样本数据
xx = 0:25:10;
yy = spline(x,y,xx);            %按照两种方法调用样条函数，绘图并比较之
subplot(1,2,1)
```

```
plot(x,y,'o',xx,yy)
pp=spline(x,y)
yy1=ppval(pp,xx);
subplot(1,2,2)
plot(x,y,'o',xx,yy1)
[breaks,c,p]=unmkpp(pp)      %将 pp 形式分解，取前三个参数
pp1=mkpp(breaks,c)           %将分解后的 pp 形式重构，输出变量为 pp1，注意与 pp
                              比较
```

输出结果如下，绘制图形如图 6-8 所示。

```
pp =
    form: 'pp'
    breaks: [0 1 2 3 4 5 6 7 8 9 10]
    coefs: [10x4 double]
    pieces: 10
    order: 4
        dim: 1
breaks =
        0     1     2     3     4     5     6     7     8     9    10
c =
    -0.0419   -0.2612    1.1446         0
    -0.0419   -0.3868    0.4965    0.8415
     0.1469   -0.5124   -0.4027    0.9093
     0.1603   -0.0716   -0.9867    0.1411
     0.0372    0.4095   -0.6488   -0.7568
    -0.1234    0.5211    0.2818   -0.9589
    -0.1684    0.1509    0.9538   -0.2794
    -0.0640   -0.3542    0.7506    0.6570
     0.1190   -0.5463   -0.1499    0.9894
     0.1190   -0.1894   -0.8856    0.4121
p =
    10
pp1 =
    form: 'pp'
    breaks: [0 1 2 3 4 5 6 7 8 9 10]
    coefs: [10x4 double]
    pieces: 10
    order: 4
    dim: 1
```

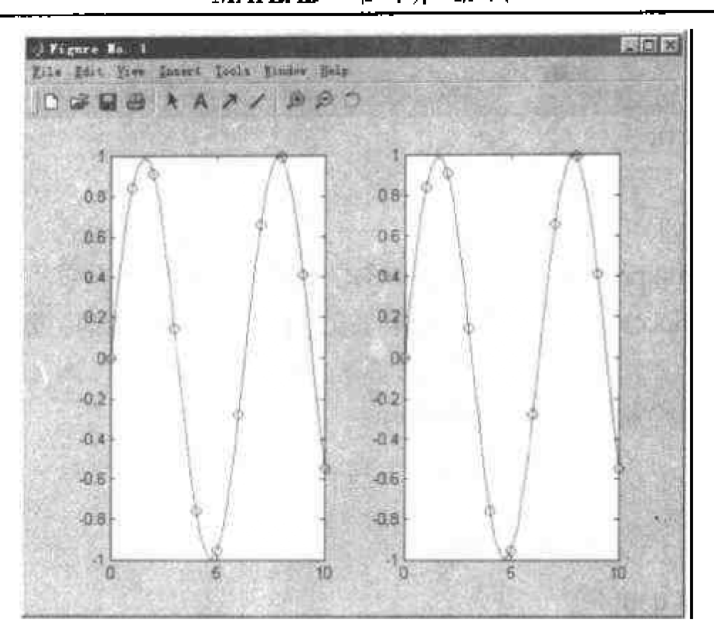


图 6-8　两种方法的插值比较图

6.5　数值积分和微分

6.5.1　一维数值积分

在工程教学和应用中，除了进行数据逼近外，还要求逼近曲线下面的面积，这就是积分问题。本节的求积运算是先利用多项式或其他函数对数据进行曲线拟合或逼近，得到数据的逼近函数，然后利用函数的数值积分函数来求某点的积分。而在对已知函数求积分时，理论上可以用牛顿-莱布尼慈公式求解，但在工程实际中并不实用，大多数函数都找不到其积分函数，有些函数的表达式非常复杂，用上述方法求解会变得相当繁琐，以至于给计算带来不便。而在对函数求积分时用数值积分就显得方便且精确，下面介绍几种求积分的方法。

在实践中，典型的数值积分方法有：用常数（0 阶多项式）近似函数矩形法；用直线（一阶多项式）近似函数曲线的梯形法；用抛物线（二阶多项式）近似函数曲线的 Simpson 法，以及用一般多项式近似函数的 Romberg 法等。表 6-8 列出了函数数值积分的一些命令。

表 6-8　　常见的一元数值积分命令

命令	含义和特点
quad	采用 Simpson 法计算积分。精度较高，较常用
quad8	采用 8 样条 Newton-Cotes 公式求数值积分。精度高，最常用
trapz	采用梯形法求定积分。计算速度快，精度差
cumtrapz	采用梯形法求一个区间上的积分曲线。计算速度快，精度差
sum	等宽矩形法求定积分。计算速度快，精度很差，一般不用
cumsum	等宽矩形法求一个区间上的积分曲线。精度很差，一般不用
fnint	利用样条函数求不定积分，与命令 spline、ppval 等配合使用。主要用于对付“表格”函数的积分

提示：本节将着重介绍 quad 和 quad8 命令。其他命令的使用方法简明易懂，这里不予介绍，用户可通过 MATLAB 的在线帮助进行了解。

MATLAB 提供的求积函数命令 quad 和 quad8 在使用时，其递推的层次限制在十层以内，达到这个限制则会提示警告信息，并且这两个函数命令都不能解决可积的奇异值问题，例如，求解 $\int_0^1 \frac{1}{\sqrt{x}} dx$。

函数 quad 和 quad8 完整的调用格式为：

- q = quad('fun',a,b,tol,trace,p1,p2,...)　　采用 Simpsou 法计算积分；
- q = quad8('fun',a,b,tol,trace,p1,p2,...)　采用八样条 Newton-Cotes 公式求数值积分。

其中，参数'fun'是被积函数，可以是表达式字符串、内联函数、*M* 函数文件名，被积函数的自变量，一般采用字母 x；a、b 分别是积分的上、下限，都是确定的值；tol 是一个二元向量，它的第一个元素用来控制相对误差，第二个元素用来控制绝对误差，缺省时积分的相对精度为 0.001；trace 如果取非零值时，将以动态图形的形式展现积分的整个过程，若取零值，则不画图，其缺省值是 0；$p1$ 和 $p2$ 是向被积函数传递的参数。

在上面的调用格式中，前三个输入参数是调用时必须的，而后面的输入参数可缺省。

【例 6-45】求函数的数值积分。

（1）建立函数 funq

```
function y=funq(x)
y=x.^3+x.^2+2;
```

（2）对被积函数 funq 进行数值积分

```
q=quad('funq',-1,1,1e-4)        %使用 quad 命令求数值积分
q =
    4.6667
q8=quad8('funq',-1,1,1e-4,1)   %用 quad8 命令求数值积分
q8 =
    4.6667
```

程序的运行结果显示出积分的过程如图 6-9 所示。

6.5.2 多重数值积分

一元函数积分中存在的问题，同样存在于多重积分中。下面介绍 $I=\int_{y1}^{y2}\left[\int_{x1(y)}^{x2(y)} f(x,y)dx\right]dy$ 的二重积分，至于三重积分或更高的积分，其处理方法相同。

1. 积分限为常数的二重积分命令

从 MATLAB 5.3 版本起，MATLAB 增加了一个新的闭型积分命令 dblquad，其最完整的调用格式为：

result = dblquad('fun',inmin,inmax,outmin,outmax,tol,method)

在此格式中，输入参数 fuu 是被积函数，可以直接用字符串内联函数或 M 函数文件表达，但不论什么形式，该被积函数应有两个变量，即内变量和外变量。内变量接受向量

输入，外变量接受标量输入。被积函数的输出是与内变量同长的向量。

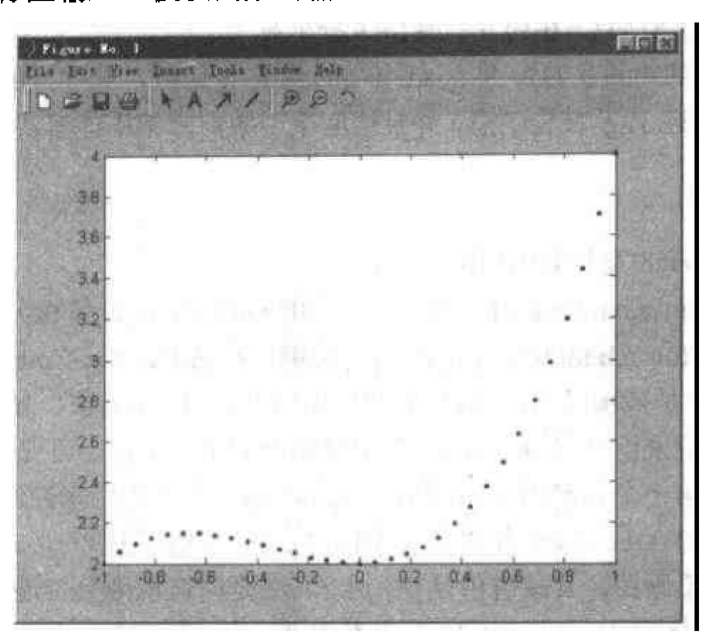


图 6-9　用图形展示积分过程

输入参数 inmin，inmax 是内变量的下限和上限；outmin、outmax 是外变量的下限和上限；tol 的含义与命令 quad 中的情况相同；method 是积分方法选项，如“quad”和“quad8”等。

注意：该命令不适用于内积分区间上下限为函数的情况。

【例 6-46】求积分上下限都为常数的二重积分，被积函数为 $y*\sin(x)+x*\cos(y)$，其中 x 的取值范围是 π 到 2π，y 的取值范围是 0 到 π。

（1）建立名为 integrnd 的 M 文件

```
function out = integrnd(x, y)
out = y*sin(x)+x*cos(y);
```

（2）用函数 dblquad 命令来求 integrnd 的二重积分

```
result = dblquad('integrnd', pi, 2*pi, 0, pi)
result =
    -9.8698
```

2. 内积分上下限为函数的二重积分

对于内积分上下限是外积分变量的函数的积分问题，求解过程较为麻烦。一般方法都是先求出 $G(y)=\int_{x_1(y)}^{x_2(y)} f(x,y)\mathrm{d}x$，然后再求 $I=\int_{y_1}^{y_2} G(y)\mathrm{d}y$。

6.5.3 数值微分

在工程试验或工程应用中，有时要根据已知的数据点，求某一点的一阶或高阶导数，这时就要用到数值微分。与积分相反，数值微分非常困难。积分描述了一个函数的整体或宏观性质，而微分则描述了一个函数在一点处的斜率，即函数的微观性质。

数值微分的基本思路是先用逼近或拟合等方法将已知数据在定范一围内的近似函数求

出，再用特定的方法对此近似函数进行微分。通常有两种方法：

1. 多项式求导发求数值微分

已知函数某些节点的值，只要将用曲线拟合得到的多项式微分，再对微分后的多项式求值，即可求出在拟合范围以内任意一点的任意阶微分。该种方法一般只用在低阶数值微分。

【例 6-47】用 5 阶多项式拟合函数 cos(x)，并利用多项式的求导来求 π 处的一阶和二阶导数。

```
x=0:0.3:4;
y=cos(x);
p=polyfit(x,y,5);          %生成拟合多项式
pp=polyder(p)              %对拟合多项式求一阶微分
pp =
     -0.0330     0.2067     0.0027    -1.0257     0.0069
polyval(pp,pi)             %求 pi 处的一阶导数
ans =
     0.0025
ppp=polyder(pp)            %求多项式的二阶微分
pp =
   -0.1321     0.6200     0.0053    -1.0257
polyval(ppp,pi)            %求 pi 处的二阶导数
ans =
     1.0150
```

2. 用 diff 计算差分求数值微分

在给定一些描述某函数的数据后，MATLAB 提供了一个计算非常粗略的微分函数命令 diff，它计算数组中元素间的差分。调用格式为：

***D*=diff(*X*)**

由于微分定义为：

$$\frac{\mathrm{d}y}{\mathrm{d}y}=\lim_{h\to 0}\frac{f(x+h)-f(x)}{(x+h)-(x)}$$

则 $y=f(x)$ 的微分可近似为：

$$\frac{dy}{dx}\approx\frac{f(x+h)-f(x)}{(x+h)-(x)}\qquad (h>0)$$

它是 y 的有限差分除以 x 的有限差分。又因 diff 计算数组或矩阵元素间的差分，所以在 MATLAB 中可近似求得函数的微分。

【例 6-48】用 diff 近似求函数的微分。

```
x=0:0.3:4;
```

```
y=cos(x);
dy=diff(y)./diff(x)   %应用数组近似计算微分
dy =
    Columns 1 through 7
    -0.1489   -0.4333   -0.6791   -0.8642   -0.9721   -0.9931   -0.9255
    Columns 8 through 13
    -0.7752   -0.5556   -0.2864    0.0084    0.3024    0.5694
```

因函数 diff 计算的是数组元素间的差分，故所得输出比原数组少了一个元素。这样，在绘制微分曲线时，必须舍弃 x 数组中的一个元素。当舍弃 x 的第一个元素时，上述过程给出向后差分近似。而舍弃 x 的最后一个元素，则给出向前差分近似。如果比较这两条曲线，就很容易发现，用有限差分近似微分会导致很差的结果，特别是在处理被噪音污染了的数据方面。

6.6　符号微积分应用

在工程教学和工程实践中，经常用到高等数学中的极限、导数、微分和积分等概念。MATLAB 的数学工具箱提供了微积分运算的基本函数：微分、积分、求和、极限和泰勒展开，本节主要介绍这些功能，它们是能否顺利、熟练地使用 MATLAB 解决实际问题的关键。

6.6.1　符号自变量的确定

在 MATLAB 中进行数学运算时，自变量的选取一般根据上下文得到。例如，在表达式 $f = x^3$ 和 $y = \sin(at + b)$ 中，如果对其进行求导而没有指定自变量，则 MATLAB 将根据数学约定，分别得到求导式：$f = 3x^2$ 和 $y = a\cos(at + b)$，即分别假设这两个表达式中的自变量为 x 和 t，其他符号 a 和 b 都看作常数或参数。

根据数学约定，自变量一般都是小写字母，并且在拉丁字母表的后面（如 x、y 和 z）。在 MATLAB 中，可以用函数 findsym 找出自变量的符号。

【例 6-49】如何确定自变量。

```
syms a b c x y
f=a*x^3+b*x-c;        %定义一个符号表达式
g=sin(x+3*y);         %定义另一个符号表达式
diff(f)               %对第一个表达式求导，但并没有说明对哪一个变量运算
ans =
    3*a*x^2+b
findsym(f,1)          %寻找f中的第一个符号变量
ans =
```

```
      x
findsym(g,1)          %寻找g中的第一个符号变量
ans =
      x
findsym(f)            %给出表达式f中的所有符号变量
ns =
    a, b, c, x
```

函数 findsym 的运算规则是在符号表达式中的缺省变量为靠近小写字母 x 的优先，在 x 后面的字母优先。

6.6.2 极限

在实际工作中，极限的求法有很多技巧，因而往往比较复杂。MATLAB 采用函数 limit 直接计算函数的极限，其调用格式为：

- limit(*f*,*x*,*a*) 求符号表达式 f 当 x ->a 时的极限；
- limit(*f*,*a*) 对系统默认变量且该默认变量-> a 时表达式 f 的极限；
- limit(*f*) 对系统默认变量且该默认变量-> a=0 时表达式 f 的极限；
- limit(*f*,*x*,*a*,'right')或 limit(*f*,*x*,*a*,'left') 求 x 从右侧或从左侧趋近 a 时表达式 f 的极限。

【例 6-50】求极限。

```
syms x h
limit((sin(x+h)-sin(x))/h,h,0)   %求当h->0时的极限
ans =
cos(x)
limit((x-2)/(x^2-4),2)           %求对默认变量->2时的极限
ans =
1/4
limit(sin(x)/x)                  %求对默认变量->0时的极限
ans =
1
limit(1/x,x,0)
ans =
NaN
limit(1/x,x,0,'right')           %x从右侧趋近0时的极限
ans =
inf
limit(1/x,x,0,'left')            %x从左侧趋近0时的极
ans =
-inf
```

对于极限 $\lim_{x \to a} f(x)$，不论从左侧趋近 a 还是从右侧趋近 a，其结果都相同，则 $f(x)$ 在 a 的极限存在；如果两侧的值不同，则说明 $f(x)$ 在 a 点的极限不存在。表达式 $f(x)$ 在其奇异点处没有极限。

从例 6-50 可知，系统缺省情况下 limit(*f*)等同于 limit(*f*,*x*,0)。

6.6.3 导数和微分

由高等数学可知，函数 *f*(*x*,*y*,*z*,...)在某一点（x_0,y_0,z_0...）的增长率就是此函数在该点的导数。对函数的严格定义为：

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

可以用前面的 limit 命令来求各种函数的导数。而在 MATLAB 中，求函数的导数或微分由专门的函数 diff 来完成。

1. 求符号表达式的导数和微分

diff 函数有四种调用格式，分别为：

- diff(*f*) 对默认的自变量求微分，并且自变量由函数 findsym 确认；
- diff(*f*,*a*) 对自变量 *a* 求微分；
- diff(*f*,*n*) 对默认的自变量求 *n* 阶微分，并且自变量由函数 findsym 确认；
- diff(*f*,*a*,*n*) 对变量 *a* 求 *n* 阶微分。

当 *a* 缺省时，自变量自动由 findsym 确认；*n* 缺省时，默认 *n*=1。

【例 6-51】符号表达式的求导与微分。

```
syms a b c x f;
f=a*x^4+x^3-b*x+c;     % 定义一个符号表达式
diff(f)                 %对默认的变量求微分，等价于 diff(f,x,1)
ans =
4*a*x^3+3*x^2-b
diff(f,a)               %对变量 a 求微分
ans =
x^4
diff(f,2)               %对默认的变量求二次微分，等价于 diff(f,x,2)
ans =
12*a*x^2+6*x
diff(f,a,3)             %对变量 a 求三次微分
ans =
0
```

2. 对符号数组或符号矩阵求导和微分

函数 diff 也可对数组进行运算。如果 *f* 是符号数组或矩阵，则求导对数组或矩阵中的每个元素进行。下面举例 6-52 来说明。

【例 6-52】符号矩阵的微分。

```
syms a b x
F=[cos(a*x) sin(a*x);-sin(b*x) cos(a^2*x)];
diff(F)                          %对 F 求微分
ans =
[       -sin(a*x)*a,        cos(a*x)*a]
[       -cos(b*x)*b, -sin(a^2*x)*a^2]
diff(F,a,2)                      %对变量 a 求二阶微分
ans =
[                       -cos(a*x)*x^2,                        -sin(a*x)*x^2]
[                       0, -4*cos(a^2*x)*a^2*x^2-2*sin(a^2*x)*x]
diff(diff(f,a),x)                %求二阶混合导数
ans =
4*x^3
```

3. 计算数值向量的数值差分

函数 diff 也可以计算数值向量或矩阵的数值差分。对于一个数值向量或矩阵 *F*，diff(*F*) 计算 *F*（2:m,:）-*F*(1:m-1,:)的数值差分。其调用格式为：

Y=diff(*F*, *n*, dim)

其中，*F* 是向量或数组；*n* 是差分阶数；dim 是指定沿着数组的哪一维进行差分。而 *n* 和 dim 可以省略。

注意：向量或数组的有限差分是按列进行的。

【例 6-53】数值向量的差分。

```
F=[(1:6).^3]          %建立一个向量
F =
     1     8    27    64   125   216
diff(F)               %计算元素之间的差分
ans =
     7    19    37    61    91
```

4. 多元向量函数的雅可比式

MATLAB 中可使用 jacobian 函数计算变换的雅可比矩阵 *J*。其调用格式为：

***J*=jacobian(*f*,*v*)**

下面通过例 6-54 来说明如何计算关于向量 *v* 的雅可比矩阵 *J*。

【例 6-54】求函数的 jacobian 矩阵。

```
syms r l f
x=r*sin(l)*sin(f);y=r*cos(l)*sin(f);z=r*cos(l);
J=jacobian([x;y;z],[r l f])        %计算雅可比矩阵
J =
   [     sin(l)*sin(f),  r*cos(l)*sin(f),  r*sin(l)*cos(f)]
   [     cos(l)*sin(f), -r*sin(l)*sin(f),  r*cos(l)*cos(f)]
```

```
[          cos(l),          -r*sin(l),                    0]
```

注意：jacobian 函数的第一个输入参数必须是列向量，第二个输入参数必须是行向量。

因 jacobian 行列式是非常复杂的三角函数表达式，所以用 simple 命令可对其进行三角变换和简化。

6.6.4 符号积分

积分有不定积分、定积分、广义积分和重积分等。一般说来，积分比微分更难求。符号积分与数值积分相比，符号积分指令简单，适应性强，但可能占用的机器时间较长。有时可能得不出“闭”解，但都给警告提示和积分原式。

在 MATLAB 中，函数 int(*f*)命令用来进行符号积分。该命令试图找到一个符号表达式，使得 dff(*F*)=*f*. 也就是说 int(*f*)返回 *f* 的不定积分，或者是 *f* 的微分的逆运算。与微分类似，int(*f*,*a*)使用符号对象 *a* 作为积分自变量。积分命令的调用格式为：

- int(*s*)　求符号表达式 *s* 关于 findsym 确定的默认符号变量进行不定积分；
- int(*s*,*v*)　求符号表达式 *s* 关于变量 *v* 的不定积分；
- int(*s*,*a*,*b*)　求符号表达式 *s* 关于默认变量从 *a* 到 *b* 的定积分，其中 *a* 和 *b* 是数值；
- int(*s*,*v*,*a*,*b*)　求符号表达式 *s* 关于变量 *v* 的从 *a* 到 *b* 的定积分。

正如函数 diff 一样，积分命令 int 对符号数组或矩阵的每一个元素进行运算。

a、*b* 分别是积分的上、下限，允许它们取任何值或符号表达式。

当积分限由一具体值变为正负无穷时，定积分便转化为广义积分，此时也只需将积分限变为无穷，即可得到相应函数的广义积分值。

【例 6-55】符号积分运算。

```
syms x u t;
A = [cos(x*t),sin(x*t);-sin(x*t),cos(x*t)];   %定义一个符号矩阵
s=sin(u+3*x);                                 %定义一个符号表达式
int(s)                                        %以默认变量 x 进行积分
ans =
-1/3*cos(u+3*x)
int(s,u)                                      %以 u 为变量进行积分
ans =
-cos(u+3*x)
int(s,pi/2,pi)                                %以变量 x 从 pi/2 到 pi 进行积分
ans =
1/3*cos(u)+1/3*sin(u)
int(s,u,pi/2,pi)                              %以变量 u 从 pi/2 到 pi 进行积分
ans =
4*cos(x)^3-3*cos(x)-4*sin(x)*cos(x)^2+sin(x)
int(s,x,2,sin(t))                   %以变量 x 从 2 到 sin(t)进行积分
ans =
-4/3*cos(u)*cos(sin(t))^3+cos(u)*cos(sin(t))+4/3*sin(u)*sin(sin(t))*cos(sin(t))^2-
```

```
1/3*sin(u)*sin(sin(t))+1/3*cos(u+6)
    int(A,t)                          %以变量 t 对矩阵 A 进行积分
    ans =
    [ 1/x*sin(x*t),   -cos(x*t)/x]
    [    cos(x*t)/x, 1/x*sin(x*t)]
    syms x k
    f=exp(-(k*x)^2);
    int(f,x,-inf,inf)
```

6.6.5 符号求和

当符号变量的和存在时，可用函数 symsum 进行符号求和。其调用格式为：

- symsum(*s*) 关于默认变量对通项 *s* 求不定和；
- symsum(*s*,*v*) 关于指定变量 *v* 对通项 *s* 求不定和；
- symsum(*s*,*a*,*b*) 或 symsum(*s*,*v*,*a*,*b*) 关于默认变量或指定变量 *v* 在[*a*，*b*]之间取值时，对通项 *s* 求和。

【例 6-56】求下面级数的和。

```
syms k n
simple(symsum(k^2,0,n))               %求和0+1^2+2^2+…+n^2
ans =
1/6*n*(n+1)*(2*n+1)
symsum(1/k^2,1,Inf)                   %求和1+1/2^2+1/3^2+…
ans =
1/6*pi^2
```

（注释中的数学式：$0+1^2+2^2+\cdots+n^2$；$1+\frac{1}{2^2}+\frac{1}{3^2}+\cdots$）

6.6.6 泰勒级数

在 MATLAB 中，提供有函数 taylor，用于对变量进行泰勒级数的展开。taylor 函数的调用命令为：

- taylor(*f*) 求 *f* 对默认变量的泰勒展开，六阶小量以余项式给出；
- taylor(*f*,*n*) 求 *f* 对默认变量的前（*n*-1）项非零泰勒展开多项式；
- taylor(*f*,*a*) 求 *f* 对默认变量在点 *a* 处的泰勒展开多项式；
- taylor(*f*,*x*) 用指定变量 *x* 代替 findsym 所确定的变量。

【例 6-57】求下面的泰勒级数。

```
taylor(exp(-x))          %求对默认变量的前 6 个非零项
ans =
1-x+1/2*x^2-1/6*x^3+1/24*x^4-1/120*x^5
taylor(log(x),8,1)       %求 log(x)的在 x=1 点处的泰勒展开式的前面 8 个非零项
ans =
```

```
x-1-1/2*(x-1)^2+1/3*(x-1)^3-1/4*(x-1)^4+1/5*(x-1)^5-1/6*(x-1)^6+1/7*(x-1)^7

taylor(x^t,3,t)          %求x^t对指定变量t的泰勒展开式的前面3个非零项

ns =
1+log(x)*t+1/2*log(x)^2*t^2
```

6.7 常微分方程的求解

在科学研究和工程教学中会经常遇到常微分方程。只含有一个自变量的微分方程称为常微分方程（Ordinary Differential Equations，ODE），它或者没有解析解，或者求取解析解的代价无法忍受，或者只有数值解等。在 MATLAB 中，对常微分方程的解法一般有两种：数值解和符号解（解析解）。

常微分方程的求解问题可分为初值问题（Initial Value Problem，IVP）和边值问题（Boundary Vaule Problem，BVP）。两者比较而言，解决边值问题的难度更大，并且类型繁多，必须具体分析具体解决。而在实际应用中，有相当一部分边值问题，最终要变换为初值问题才能解决。

6.7.1 常微分方程的数值解法

当微分方程式能够解析求解时，可用 MATLAB 符号工具箱中的功能找到精确解，但是获得解析解一般很困难。在微分方程难以获得解析解的情况下，可以方便地在数值上求解。微分方程数值解是数值计算的基本内容，由于微分方程的多样性，因而它也有很多不同的解法。MATLAB 给出的 7 种解法分别由 7 个不同的函数来完成。

1. 刚性（Stiff）问题

在介绍解常微分方程的命令函数之前，先介绍微分方程的刚性问题，对于一个常微分方程组，如果其 Jacobian 矩阵的特征值相差十分悬殊，那么这个方程组就称为刚性方程组。对于刚性方程组，为保持解法的稳定，步长选取很困难。有些解法不能用来解刚性方程组，而有些解法对稳定性的要求不严格，可以用来解决刚性问题。对于 MATLAB 提供的几种解法的解体类型，将在下面详细地介绍。

2. *初值常微分方程解算有关的命令及特点*

MATLAB 为解决常微分方程初值问题提供了一组配套齐全、结构严整的命令，包括微分方程结算命令、被解算命令、调用的常微分方程文件格式命令、积分算法参数选项（Options）处理命令以及输出处理命令等。关于这些命令的含义、特点和使用范围等如表 6-9 所示。

表中的单步法是指只需要前一步的解即可计算出当前的解，不需要附加初始值。在计算过程中可随便改变步长而不会增加任何附加的计算量。

表中的多步法是指需要前几次的解来计算当前的解。

表 6-9　**初值常微分方程解算命令表**

命令		含义	解题类型	特点	适用场合
解算命令	ode23	普通 2、3 阶法解 ODE	非刚性	属于单步法；采用 2、3 阶龙-库塔（Runge-Kutta）法，累计截断误差（△ x）^{3}	较低精度（10^{3}）场合
	ode45	普通 4、5 阶法解 ODE	非刚性	单步法；采用 4、5 阶龙格-库塔法，累计截断误差达（△ x）^{5}	大多数场合的首选算法
	ode113	普通变阶法解 ODE	非刚性	多步法；采用 Adams-Bashforth-Moulton PECE 算法，高低精度均可（10^{3}~10^{6}）	ode45 计算时间太长，用其取代 ode45
	ode23t	解适度刚性 ODE	适度刚性	采用自由内插法实现的梯形规则（Trapezoidalrule)法	适度刚性且要求无数值衰减的情况
	ode15s	变阶法解刚性 ODE	刚性	多步法：采用 Gear's 反向数值微分算法；精度中等	当 ode45 失败时使用；或存在质量矩阵时
	ode23tb	低阶法解刚性 ODE	刚性	采用 TR-BDF2 法实现，即梯形规则-反向数值微分（Gear）两阶段算法；低精度	低精度时，比 ode15s 有效；或存在质量矩阵时
odefile		常微分方程（ODE）文件格式			
选项	odeset	创建、更改常微分方程选项的设置			
	odeget	读取 ODE 选项的设置			
输出	odeplot	ODE 的输出时间序列图			
	odephas2	ODE 的二维相平面图			
	odephas3	ODE 的三维相平面图			
	oodeprint	在 MATLAB 命令窗口显示结果			

3. MATLAB 解常微分方程的一般步骤

对于 n 阶微分方程初值问题，由于函数及其直至（n-1）阶导数在某自变量点的值已知，所以由泰勒级数展开，可算出新的函数及导数值。在 MATLAB 中，具体利用其命令来解初值问题。常微分方程的步骤是：

（1）根据在工程实际中各学科的规律、定理和公式列出微分方程和相应的初始条件。

（2）运用变量替换，把一个高阶方程写成一阶微分方程组，初始条件也要做相应地替换。

（3）根据变换后的一阶微分方程组，编写计算导数的 M 文件（在 MATLAB 中称之为 ODE 文件）。

（4）使编写好的 ODE 函数文件和变换后的初值供 Y_0 微分方程解算命令（如表 6-9 中所列）调用，运行后即可得到 Y（包含 y 及其导数）在指定的时间区间上的数值解。

4. 解算微分方程的命令调用格式

MATLAB 提供了多种解算常微分方程的命令（如表 6-9 中所列），这些命令函数的用

法完全相同，以函数 ode45 为例，其最完整的调用格式为：

[*T*,*Y*] = ode45('*f* ',tspan,*y*0,options,*p*1,*p*2,...)

式中，'*f* '是定义常微分方程的文件名字符串，这些函数是针对一阶常微分方程组设计的，对于高阶微分方程，必须先化为形如 $\dot{x} = f(x,t)$ 的一阶微分方程组。在这里要注意：定义常微分方程组的函数必须以 t、x 为输入参数（顺序不能变），以 $\dot{x}$ 为输出参数，并且输出 $\dot{x}$ 的必须是列向量。

tspan 是一个向量，当被赋予二元向量[t0　tfinal]时，指定微分方程 $y' = F(t,y)$ 在初始条件 *y*0 下，从 *t*0 到 tfinal 进行积分；当 tspan 被赋予多元向量[*t*0 *t*1 ... *t*final]时，在 tspan 指定的时刻序列上求数值解。此时 tspan 中的元素必须按单调次序排列。

*y*0 是初始状态列向量。$y0=\begin{bmatrix} y_0 \\ y'_0 \\ \vdots \\ y^{(n-1)}{}_0 \end{bmatrix}$

options 是一些可选的算法综合参数，由函数 odeset 进行设置。当输入参数只有三个时，或第四个输入参数位置上用空矩阵“[]”赋值时，算法将使用 options 的缺省设置。用户可参看 MATLAB 的在线帮助来了解其用法。

*p*1 和 *p*2 是传递给'*F*'的参数，从第 5 个输入参数起，以后的各输入参数都是向 ODE 文件传递的参数。它们的使用可避免采用参数的全局变量方式传递。

T 是输出参数，为所求数值解的时间列向量，*Y* 是数值解的矩阵，它的每一行对应着列向量 T 中的一个时间值。

说明：调用格式中的参数'*f*'、tspan 和 *y*0 是必须的，其他参数可选。

【例 6-58】解常微分方程组（非刚性问题）。

已知常微分方程 $\frac{d^2x}{dt^2} - w(1-x^2)\frac{dx}{dt} + x = 0$ 并且初始条件为 $x_0 = 1, x'_0 = 0$。

（1）与所有的数值求解微分方程一样，高阶微分方程式必须等价地变换成一阶微分方程组。对于上述微分方程，通过重新定义两个新的变量来实现这种变换。

令 *y*(1)=*x*，则 *y*(2)=d*y*(1)/d*t*

d*y*(2)/d*t*=*w*(1-*y*(1)^2)**y*(2)-*y*(1)

（2）编写一个该方程组的描述 M 文件 ode1.m，给定当前时间及 *y*1 和 *y*2 的当前值，该函数返回上述导数值。在 MATLAB 中，这些导数值由一个列向量给出。而在本例中，这个列向量为 **dy**，同样，*y*1 和 *y*2 合并写成列向量 *y*。即所描述的 M 文件为：

```
function dy=ode1(t,y)        %t 和 y 的先后顺序不能变
dy=zeros(2,1);               %首先定义 dy 为两个零元素的列向量
dy(1)=y(2);                  %输入微分方程表达式
```

```
dy(2)=2*(1-y(1)^2)*y(2)-y(1);
```

解方程组。因此方程组为非刚性问题，因而选 ode45 算法。

```
[T,Y]=ode45('ode1',[0 30],[1;0]);     %在区间 0 到 30 时间段上求解
y1=Y(:,1);      %第一列向量，速度
y2=Y(:,2);      %第二列向量，位移
plot(T,y1,'*',T,y2,'-r')              %分别用不同的线型绘制维分方程的解
legend('速度','位移')
```

程序的执行结果如图 6-10 所示。

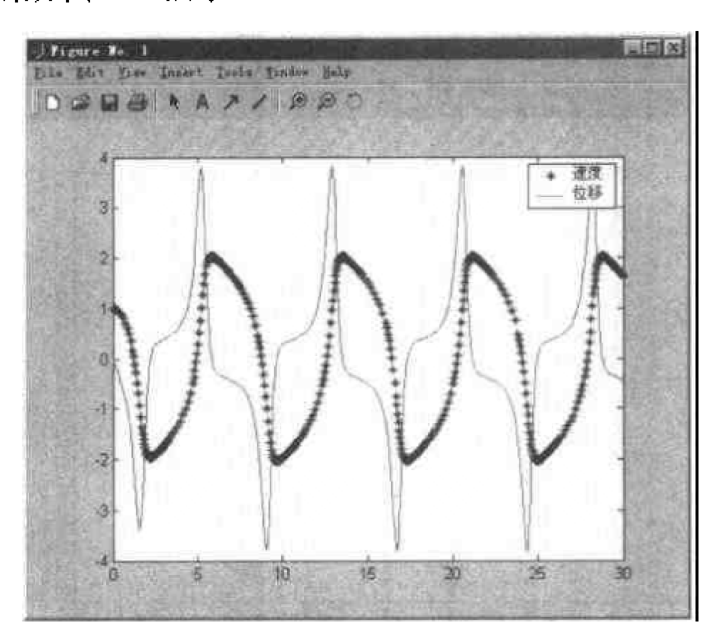


图 6-10 非刚性微分方程的解

【例 6-59】解微分方程组（刚性问题）。

已知二阶微分方程 $y''=1000(1-y^2)y'-y$，初始条件：$y(0)=0$，$y'(0)=1$，求该微分方程的解。

为求该方程的解，则应按下面步骤进行：

（1）将该方程降为一阶微分方程组

令 $y_1=y, y_2=y_1'$，则该方程可化为如下的一阶微分方程组：

$$\begin{cases} y_1'=y_2 \\ y_2'=1000(1-y_1^2)y_2-y_1 \end{cases}$$

（2）建立该方程组的描述文件 ode2.m

```
function dy=ode2(t,y)          %t 和 y 的先后顺序不能变
dy=zeros(2,1);
dy(1)=y(2);                    %输入微分方程的表达式
```

```
dy(2)=1000*(1-y(1)^2)*y(2)-y(1);
```

（3）解方程组

因为这是一个刚性问题，所以选用 ode15s 算法，其求解区间设为[0 3500]，并用函数 odeset 设置相对误差容限 1e-4，绝对误差容限 y_1、y_2 分别为 1e-4 和 1e-5。其命令如下：

```
options=odeset('RelTol',1e-4,'AbsTol',[1e-4,1e4-5]);
[T,Y]=ode15s('ode2',[0 3500],[0;1],options);
plot(T,Y(:,1),'-')        %矩阵 Y 的第一列是原二阶微分方程的解
```

其执行结果如图 6-11 所示。

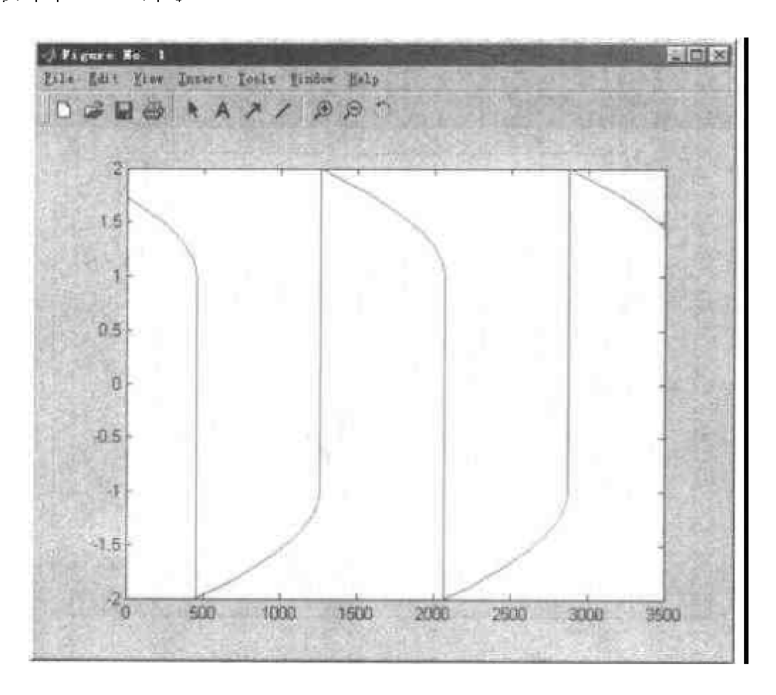


图 6-11 刚性微分方程的解

6.7.2 MATLAB 中 ODE 文件说明

在 6.7.1 的例题中，对 ODE 文件的具体应用作了介绍。为使用户对 ODE 文件有更为全面的理解，下面将对 ODE 文件作系统地介绍。

ODE 文件是从 MATLAB 5.x 版本起专门被微分方程解算命令（如 od45）调用的 M 函数文件。ODE 文件的功能很多，其中计算每一步的导数是其最基本、必不可少的功能。另外，还有一些常用的功能，如解算指令中 options 的参数设置、事件设置等。

MATLAB 给 ODE 文件制定了比较严整的格式和规定。而对于初学者来说，这样的 ODE 文件就显得比较复杂。为帮助用户理解和编写 ODE 文件，MATLAB 提供了一个"模板"文件 odefile.m。用户在自己编写 ODE 文件时，可以借助此模板，在模板中填入适当的内容，删去不需要的内容。

1. ODE 文件模板

一个 ODE 文件就是被微分方程解算指令（如 od45）调用的 M 函数文件，用户可根据需要，使用任何其他文件名。

在缺省情况下，ODE 解算指令处理形如 $dy/dt = F(t,y)$的微分方程问题，t 是自变量，y

是一阶微分方程组的列向量形式函数。解算时，解算指令重复调用 F =odefile(T,Y)，此处 T 是一个时间序列，Y 是一个列向量，输出参数 F 是一个同长度的列向量。ODE 文件的经典最简形式为：

```
function F = odefile(t,y)
F = < Iusert a fuuction of t and/or y here. >;
```

在 ODE 文件中，最完整的格式为：

```
functiou varargout = odefile(t,y,flag,p1,p2...)
```

在此式中，t 和 y 是积分变量，t 必须作为第一输入参数，而 y 必须作为第二输入参数。$p1$ 和 $p2$ 是被传递的参数，这里仅列出两个参数，实际上，传递参数的数目不受限制。flag 是切换变量，它必须处在第三个输入参数位置上，其具体用法为：

flags	返回值
''(empty)	$F(t,y)$
'init'	tspan, y0 和 options 取缺省值
'jacobian'	雅可比矩阵为 $J(t,y) = dF/dy$
'jpattern'	显示雅可比矩阵的稀疏方式
'mass'	用于解方程 $M*y' = F(t,y)$的质量矩阵 M、$M(t)$或 $M(t,y)$
'events'	事件定义和判断

```
switch flag
case ''                %规定空字符串''情况，专管计算 dy/dt = f(t,y)
    varargout{1} = f(t,y,p1,p2);
case 'init'            %规定'int'情况，专管三个输入参数 [tspan,y0,optious].的设置
    [varargout{1:3}] = init(p1,p2);
case 'jacobian'        %规定'jacobian'情况，专管计算解析雅可比矩阵的 df/dy
    varargout{1} = jacobian(t,y,p1,p2);
case 'jpattern'        %规定'jpattern'情况，专管计算稀疏的数值 jacobian 矩阵 S
    varargout{1} = jpattern(t,y,p1,p2);
case 'mass '           %规定'mass'的情况，专管计算质量矩阵
    varargout{1} = mass(t,y,p1,p2);
case 'events'     %规定'events'情况，专管事件的定义和判断 [value,isterminal,direction]
    [varargout{1:3}] = events(t,y,p1,p2);
otherwise
    error(['Unknown flag ''' flag '''.']);
end
%以下是 odefile.m 的子函数，它们分别与主函数中各情况对应
%-----------------------------------------------------------------------
function dydt = f(t,y,p1,p2)
%“空字符”情况的调用，专管一阶导数 dy/dt 的计算
%dydt 是一阶导数，该变量名可由用户按需要取名
%下面是函数内容，由用户自己编写，最后产生输出 dydt
```

```
dydt = < Insert a function of t and/or y, p1, and p2 here. >
 % ---------------------------------------------------------------------------
function [tspan,y0,options] = init(p1,p2)
%'int'情况下调用的子函数，专管三个参数[tspan,y0,options]
%三个输出参数的名字可由用户自己起，但各位置上参数的性质不能改变
%下面由用户自己编写，最后产生三个输出参数
tspan = < Insert tspan here. >;
y0 = < Insert y0 here. >;
options = < Insert options = odeset(...) or [] here. >;
% ---------------------------------------------------------------------------
fuuction dfdy = jacobian(t,y,p1,p2)
%'jacobian'情况下调用的子函数，专管计算解析的雅可比矩阵 df/dy
%输出参数的名字可由用户自己起
% 下面的内容可由用户自己编写，最后产生输出参数 dfdy
dfdy = < Insert Jacobian matrix here. >;
% ---------------------------------------------------------------------------
function S = jpattern(t,y,p1,p2)
%'jpattern'情况下调用的子函数，专管计算解析雅可比矩阵 df/dy
%输出参数的名字可由用户自己起
%下面的内容可由用户自己编写，最后产生输出参数 S
S = < Insert Jacobian matrix sparsity pattern here. >;
% ---------------------------------------------------------------------------
function M = mass(t,y,p1,p2)
%'mass'情况下调用的子函数，专管计算质量矩阵，解刚性方程时用
% 输出参数的名字可由用户自己起
%下面的内容可由用户自己编写，最后产生输出参数 M
M = < Insert mass matrix here. >;
% ---------------------------------------------------------------------------
function [value,isterminal,direction] = events(t,y,p1,p2)
%'events'情况下调用的子函数，专管三个参数[value,isterminal,dirctiotı]的设置
%输出参数的名字可由用户自己起
%下面的内容可由用户自己编写，最后产生输出三个输出参数
value = < Insert event function vector here. >
isterminal = < Insert logical ISTERMINAL vector here.>;
direction = < Insert DIRECTION vector here.>;
```

2. ODE 模板的使用方法

如果用户需要编写比较复杂的 ODE 文件，该模板将十分有用。其具体步骤如下：

（1）在 MATLAB 命令窗口中运行 help odefile，就会出现 ODE 模板。

（2）将此模板文件全部内容复制到文件编辑器中。

（3）用户根据需要，将模板文件中的那些不被使用的情况（Case）连同相应的子函数删除。

如无特别原因，尽量少改动模板中所有的子函数名、输入和输出参数名。

用户根据具体问题编写相应的程序。

（4）把编写好的文件存入用户自己的工作目录，以备调用。编写好的ODE文件用户可以给一个新名字，但要注意以下三者间的一致：存放在工作目录上的名称；函数文件第一行中等式右边的函数名称；在微分方程解算指令调用该ODE文件时的字符串名称。

3. MATLAB提供的微分方程帮助资源

MATLAB为微分方程指令的使用提供了丰富的帮助资源。它们的分布形式和位置为：

- PDE文件

matlabR12\help\pdf_doc\matlab\using_ml.pdf

matlabR12\help\pdf_doc\matlab\ref\refbook.pdf

- 在线帮助

使用help funfun以及对具体命令进一步使用help；或使用helpwin查阅。

MATLAB实现ODE演示的M文件

matlabR12\toolbox\matlab\demos

6.7.3　常微分方程的符号解

1. 微分方程的符号解法和数值解法互补

从数值计算角度看，与初值问题求解相比，微分方程边值问题的求解显得更为复杂和困难。而这一问题对于符号计算来说，不论是初值问题，还是边值问题，其求解微分方程的命令形式相同，并且相当简单。

但符号计算可能花费较多的时间，也可能得不到简单的解析解以及得不到封闭式的解，甚至也可能无法求解。不论怎样，微分方程的符号解法和数值解法有很好的互补作用。

2. 微分方程符号解的一般命令

在MATLAB中，函数dsolve可用来求常微分方程的符号解。在符号方程中，用符号表达式中包含的字母“D”来代替微分运算，符号D2、D3、…DN分别对应于第二、第三、……第N导数。这样，D2y就是$\mathrm{d}^2y/\mathrm{d}t^2$在符号数学工具箱中对应的形式。因变量是位于D后面的变量，缺省的自变量为t。还可以在命令中加入包含自变量的参数，将自变量变为其他符号变量。函数dsolve的调用格式为：

S=dsolve('eqn1','eqn2', ...)

输入参数包含三部分内容：微分方程、初始条件和指定独立变量。其中，微分方程是必不可少的输入内容，其余内容视需要而定，可有可无。输入的参数必须以字符形式编写。

关于指定独立变量的规定：若要指定独立变量，则总是由全部输入参数'eqn1','eqn2', ...中的最后一个参数'eqnn'定义。若不对独立变量加以专门定义，则该指令将默认小写英文字母t为独立变量。

微分方程的记述规定：当y是因变量时，用“Dny”表示“y的n阶导数”。如：

Dy 表示形如 $\frac{dy}{dt}$ 的 y 的一阶导数；Dny 表示形如 $\frac{d^n y}{dt^n}$ 的 y 的 n 阶导数。

关于初始条件的规定：初始条件或边界条件写成'y(a)=b'或'Dy(a) = b'等。a、b 可以是变量使用符以外的其他字符。当初始条件少于微分方程数时，在所得解中将出现任意常数符 C1，C2，…解中任意常数符的数目等于所缺少的初始条件数。

输出参数可有可无。当无输出参数时，在 MATLAB 工作内存中在 y1，y2，…定义的输出参数中保存计算结果。在本调用格式中，S 是结构数组。

【例 6-60】求 $\frac{dx}{dt}=-ax$ 在无初始条件和有初始条件下的解。

```
dsolve('Dx = -a*x')
ans =
C1*exp(-a*t)
x = dsolve('Dx = -a*x','x(0) = 1','s')
x =
exp(-a*s)
```

在第一个微分方程中，t 是默认变量。而在第二个微分方程中，指定独立变量为 s。C1 为任意常数。

【例 6-61】求，$\frac{df}{dt}=f+g$，$\frac{dg}{dt}=-f+g$，$f(0)=1$,，$g(0)=2$ 的解。

```
S = dsolve('Df = f + g','Dg = -f + g','f(0) = 1','g(0) = 2')
S =
    f: [1x1 sym]
g: [1x1 sym]
S.f
ans =
      exp(t)*(cos(t)+2*sin(t))
S.g
ans =
      exp(t)*(-sin(t)+2*cos(t))
```

【例 6-62】求，$w(0)=1$，$\frac{d^3 w}{dt}=-w$，$w'(0)=0$，$w''(0)=0$ 的解。

```
w = dsolve('D3w = -w','w(0)=1, Dw(0)=0, D2w(0)=0')
w =
    1/3*exp(-t)+2/3*exp(1/2*t)*cos(1/2*3^(1/2)*t)
```

【例 6-63】求 $\frac{dy}{dt}=y^2(1-y)$ 的解。

```
Y = dsolve('Dy = y^2*(1-y)')
```

Warning: Explicit solution could not be found; implicit solution returned.

> Iu D:\MATLABR11\toolbox\symbolic\dsolve.m at line 292

Y =

t+1/y-log(y)+log(-1+y)+C1=0

函数 dsolve 命令求解微分方程时，如果得不到其解，则给出警告信息。

6.8　数据分析函数和傅立叶变换

MATLAB 的基本数据分析函数位于目录\toolbox\matlab\datafun 下。MATLAB 对这些函数有两条规定：

- 若输入的参数是向量，则不论是行向量还是列向量，运算都是对向量整体进行的；
- 若输入的参数是数组（包括矩阵），则运算是按列进行的。

由于 MATLAB 是面向矩阵的语言，所以这两条规定不仅仅适用于这些数据分析函数，而且对 MATLAB 各工具箱的函数具有普遍意义。因此，用户在自己编程时，请尽量遵循以上规定，以便更好地利用 MATLAB 现有的函数和命令。

在 MATLAB 中的数据分析函数是按矩阵列向量而进行的，所以不同的变量存储在各列中，而每行表示每个变量的不同观察。MATLAB 数据分析函数如表 6-10 所示。

表 6-10　数据分析函数

函数	功能简介	函数	功能简介
corrcoef(x)	求相关系数	max(x),max(x,y)	最大分量
cov(x)	协方差矩阵	mean(x)	均值或列的平均值
cplxpair(x)	把向量分类为复共轭对	median(x)	找出列的中值
cross(x,y)	向量的向量积	min(x),min(x,y)	最小分量
cumprod(x)	列累计积	prod(x)	列元素的积
cumsum(x)	列累计和	rand(x)	均匀分布随机数
del2(A)	五点离散拉氏算子	randn(x)	正态分布随机数
diff(x)	计算元素之间的差	sort(x)	把数组的列元素按升序排列
dot(x,y)	向量的点积	std(x)	求列的标准差
gradient(Z,x,y)	近似梯度	subspace(A,B)	两个子空间的夹角
histgram(x)	直方图和棒图	sum(x)	各列的元素求和
trapz(x)	梯形数值积分	tsearch(x)	在封闭的德洛内三角中搜索
var(x)	求方差	voronoi(x)	沃龙诺依图
inpolygon(x)	检测指定点是否在多边形区域	perms	求向量按所有可能的变换组合出的矩阵
convhull(x)	返回位于凸包上点的序号	factor(x)	计算素数因子
primes(x)	列出小于或等于某个数值的所有素数	sortrows(x)	把数组的元素以某一列为准，按递增顺序排列，缺省是第一列

6.8.1 数据分析函数的基础运算和有限差分

数据分析函数的基础运算很多，用法简单，表 6-10 已列出这些函数的名称和功能。对这些函数的用法，在这里仅举几个例子加以说明，大部分函数的用法请用户在使用时参看在线帮助。

1. 数据分析函数的基础运算

【例 6-64】数据分析函数的基础运算举例。

```
A=rand(5,5);            %产生 5 行 5 列的矩阵 A
AMAX=max(A)             %求矩阵 A 各列的最大值
AMAX =
         0.9501    0.8214    0.9218    0.9355    0.8132
AAMAZ=max(AMAX)   %求矩阵 A 中的最大值
AAMAZ =
         0.9501
AMED=median(A)          %求矩阵 A 各列的中位元素
AMED =
         0.6068    0.4565    0.7382    0.8936    0.1389
AMEAN=mean(A)           %求矩阵 A 各列元素的平均值
AMEAN =
         0.6331    0.5006    0.6487    0.7124    0.2745
ASTD=std(A)             %求矩阵 A 各列元素的标准差
ASTD =
         0.2963    0.3197    0.2861    0.2783    0.3285
ACUMS=cumsum(A)         %求矩阵 A 各列的累积和
ACUMS =
         0.9501    0.7621    0.6154    0.4057    0.0579
         1.1813    1.2186    1.4074    1.3412    0.4108
         1.7881    1.2371    2.3292    2.2581    1.2239
         2.2741    2.0585    3.0674    2.6684    1.2338
         3.1654    2.5032    3.2437    3.5620    1.3727
ASORT=sortrows(A,2)     %以第二列为基准，按递增顺序排列
ASORT =
         0.6068    0.0185    0.9218    0.9169    0.8132
         0.8913    0.4447    0.1763    0.8936    0.1389
         0.2311    0.4565    0.7919    0.9355    0.3529
         0.9501    0.7621    0.6154    0.4057    0.0579
         0.4860    0.8214    0.7382    0.4103    0.009
```

2. 拉普拉斯微分算子

MATLAB 中的函数 del2 类似于离数拉普拉斯微分算子，其一般调用用法为：

L = del2(U)

其中，U 是数组，如果把 U 看作多元函数 u（$x,y,z,\ldots$），数组的维数就是函数中变量的个数，则函数 del2 相当于进行如下运算：

$L=\frac{\nabla^2 u}{2N}=\frac{1}{2N}(\frac{\mathrm{d}^2 u}{\mathrm{d}x^2}+\frac{\mathrm{d}^2 u}{\mathrm{d}y^2}+\frac{\mathrm{d}^2 u}{\mathrm{d}z^2}+\cdots)$ ，式中的 N 就是数组的维数，也就是函数中的变量个数。

【例 6-65】拉普拉斯微分算子。

```
U=[1 0 1;3 1 2;1 1 5];
L=del2(U)
L =
   -0.5000    0.2500    1.0000
   -0.2500    0.5000    1.2500
         0    0.7500    1.5000
```

3. 数值梯度

在 MATLAB 中，函数 gradient 用来进行数值梯度的计算。一般情况下，函数 $F(x,y,z,\ldots)$ 的梯度定义为：

$$\nabla F=\frac{\partial F}{\partial x}\vec{i}+\frac{\partial F}{\partial y}\vec{j}+\frac{\partial F}{\partial z}\vec{k}+\cdots$$

在 MATLAB 中，函数 gradient 的调用格式为：

[*FX*,*FY*,*FZ*] = gradient(*F*)

如果数组是一维向量，则只返回 FX，即 $\frac{\partial F}{\partial x}$；如果是二维矩阵，则返回 FX 和 FY，即 $\frac{\partial F}{\partial x}$ 和 $\frac{\partial F}{\partial y}$，分别对应矩阵的列方向和行方向，其余依此类推。

【例 6-66】求数值梯度。

```
A=[2 3 4;2 0 4;5 1 1];
[FX,FY]=gradient(A)
FX =
     1     1     1
    -2     1     4
    -4    -2     0
FY =
         0   -3.0000         0
    1.5000   -1.0000   -1.5000
    3.0000    1.0000   -3.0000
```

4. 向量运算

在实际工程数学和计算中，经常遇到向量的运算。MATLAB 提供了功能比较强大的

运算能力，下面简要介绍其用法。

- 向量的点积

在 MATLAB 中，函数 dot 用来计算向量的点积，其最完整的调用格式为：

C=dot(A,B,dim)

在此式中，如果 A 和 B 都是向量，则返回 A 和 B 的标量积，其中，A 和 B 必须长度相同。如果 A 和 B 是多维数组，标量积沿着 A 和 B 第一个长度非 1 的维进行，其中，A 和 B 必须有相同的大小。参数 dim 可选，如果指定了该参数，则表示在第 dim 维上计算 A 和 B 的标量积。

- 向量的叉积

在 MATLAB 中，函数 cross 用来计算向量的叉积，其调用格式为：

C = cross(A,B,dim)

在此式中，如果 A 和 B 是向量，则返回 A 和 B 的叉积，其中 A 和 B 必须是长度为 3 的向量。如果 A 和 B 是多维数组，则叉积沿着 A 和 B 第一个长度为 3 的维进行，其中 A 和 B 必须有相同的大小。参数 dim 是可选，如果指定了该参数，则表示在第 dim 维上计算 A 和 B 的叉积，其中 A 和 B 必须有相同的大小，而且 size(A,dim) 和 size(B,dim)必须等于 3。

【例 6-67】向量点积和叉积的计算。

```
A=[3 2 6];B=[0 7 4];
C1=dot(A,B)        %计算向量A和B的点积
C1 =
    38
C2=cross(A,B)      %计算向量A和B的叉积
C2 =
    -34    -12     21
```

5. 协方差矩阵和相关阵

协方差矩阵及其相关阵在概率和数理统计中有广泛而重要的应用，MATLAB 系统提供了下面的函数命令来实现计算协方差矩阵及其相关阵：

- C=cov(X)　如果输入参数 X 是向量，则返回值 C 是该向量的方差；如果 X 是矩阵，则它的每一列相当于一个变量，返回值 C 就是该矩阵的列与列之间的协方差矩阵；
- C=diag(cov(X))　如果 X 是矩阵，则返回值 C 就是该矩阵每个列向量的方差；
- C=sqrt(diag(cov(X)))　计算 X 的标准协方差；
- C=cov（X，Y）　X 和 Y 是同维向量或矩阵，相当于 cov([X(:) Y(:)])；
- S=corrcoef(X)　计算输入矩阵 X 的相关系数矩阵 S，X 的每一列相当于一个变量。对于相同的输入矩阵 X，相关系数矩阵 S 和协方差矩阵 C 之间的关系为：

$$S(i,j)=\frac{C(i,j)}{\sqrt{C(i,i)C(j,j)}}$$

- S=corrcoef(X,Y)　相当于 corrcoef([X Y])。

【例 6-68】计算协方差矩阵和相关阵。

```
X=[3 4;1 5];Y=[0 5 ;6 3];
CX=cov(X)       %计算 X 的协方差阵
CX =
    2.0000    -1.0000
   -1.0000     0.5000
CY=cov(Y)
CY =
    18    -6
    -6     2
CXY=cov(X,Y)
CXY =
    2.9167    -1.5000
   -1.5000     7.0000
SX=corrcoef(X)
SX =
     1    -1
    -1     1
SY=corrcoef(Y)
SY =
     1    -1
    -1     1
SXY=corrcoef(X,Y)
SXY =
    1.0000    -0.3320
   -0.3320     1.0000
```

6.8.2 傅立叶变换和傅立叶逆变换

在工程实践中，会经常遇到傅立叶变换。由于傅立叶变换有明确的物理意义，即变换域反映了信号包含的频率内容，因此傅立叶变换在信号处理和系统动态特性研究中起着重要的作用。下面介绍该种变化的计算机实现方法。

1. 傅立叶快速离散变换

离散序列的离散傅立叶变换(Discrete Fourier Transform，DFT)和傅立叶逆变换(Invense Discrete Fourier Transform，IDFT)在数学上的定义和在 MATLAB 软件中的定义基本相同，不同之处只是在于：MATLAB 中的序列或向量元素下标从 1 开始，而不是数学上的从 0 开始。因此在 MATLAB 中，它们的相应表达式为：

$$X(k)=\sum_{n=1}^{N}x(n)e^{-j(2\pi/N)(n-1)(k-1)} \qquad n=1,\cdots,N$$

$$x(k)=\frac{1}{N}\sum_{n=1}^{N}x(n)e^{-j(2\pi/N)(n-1)(k-1)} \qquad n=1,\cdots,N$$

MATLAB 提供了 fft(内置函数)、ifft、fft2、ifft2、fftn、ifftn、fftshift、ifftshift 等函数，用来计算矩阵的离散快速傅立叶变换。在数据长度是 2 次幂时，采用基-2 算法进行计算，计算速度会显著增加，因此，在使用时尽量使数据长度为 2 的幂次或者用零来填补数据。下面分别介绍在 MATLAB 中这些用于快速傅立叶变换函数的用法。

- 函数 fft 和 ifft

函数 fft 最完整的调用格式为：

***Y*=fft(*X*,[],dim) 或 *Y*= fft(*X*,*n*,dim)**

在此式中，输入参数 *X* 可以是向量、矩阵或高维数组。如果 *X* 是向量，则采用快速傅立叶变换算法做 *X* 的离散傅立叶变换；如果 *X* 是矩阵，则计算矩阵每一列的傅立叶变换；如果 *X* 是多维数组，则对第一个非单元素的维进行计算。

输入参数 *n* 以限制 *X* 的序列长度。如果 *X* 的长度小于 *n*，则用 0 来补足，如果 *X* 的长度大于 *n*，则去掉长出的部分。*n* 可以缺省，缺省时，被变换序列的长度默认为 *n*。

输入参数 dim 是在指定的维上进行操作。当 *X* 是向量时，dim 可以缺省；当 *X* 是矩阵时，dim 将用来指定变换的实施方向，即 1 指明变换按列进行，2 指明变换按行进行，缺省时变换默认按列进行；当 *X* 是高维数组时，dim 用来指定变换的实施方向。

函数 ifft 的调用格式完全同函数 fft。

和函数 fft2 和 ifft2 类似，函数 fftn 和 ifftn 对数据做多维快速傅立叶变换，相关内容用户可参看 MATLAB 在线帮助。

【例 6-69】函数 fft 的用法。

```
Y=fft(X)
Y =
              1.3618              1.1759              0.5404
   -0.0655 - 0.7238I  -0.0586 - 0.6957I  -0.0619 - 0.0035i
   -0.0655 + 0.7238I  -0.0586 + 0.6957I  -0.0619 + 0.0035i
```

- 函数 fft2 和 ifft2

函数 fft2 和 ifft2 是对数据做二维快速傅立叶变换和逆傅立叶变换。数据的二维傅立叶变换 fft2(*X*)相当于 fft(fft(*X*)')'，即先对 *X* 的列做一维傅立叶变换，然后对变换结果的行做一维傅立叶变换。其调用格式为：

***Y*=fft2(*X*,mrows,ncols)**

二维快速傅立叶变换。通过截断或用 0 补足，使得 *X* 为 $m\times n$ 的矩阵。

在此式中，输入参数 mrows 和 ncols 可以省略。

【例 6-70】函数 fft2 的用法。

```
X=rand(3,3);
Y=fft2(X)
Y =
   5.9464   -0.1951 + 0.3653i   -0.1951 - 0.3653i
```

```
-0.3149 - 0.3499i  -0.6386 - 0.4489i   0.1765 + 1.2573i
-0.3149 + 0.3499i   0.1765 - 1.2573i  -0.6386 + 0.4489i
```

● 函数 fftshift 和 ifftshift

函数 fftshift 用于把傅立叶变换结果（频域数据）中的直流分量（频率为 0 处的值）移到中间位置。其调用格式为：

***Y*=fftshift(*X*)**

在此式中，输入参数 X 是傅立叶变换结果（频域数据）。如果 X 是向量，则交换 X 的左右半边；如果 X 是矩阵，则交换其一、三象限和二、四象限；如果 X 是多维数组，则在数组的第一维交换其"半空间"。

函数 ifftshift 相当于把 fftshift 函数的操作逆转，用法相同。

【例 6-71】函数 fftshift 的用法。

```
X=rand(3,3);
Y=fft(X)
Y =
            1.7881             2.1394             1.2964
   0.5311 + 0.3254i  -0.3407 - 0.1119i   0.0365 + 0.6953i
   0.5311 - 0.3254i  -0.3407 + 0.1119i   0.0365 - 0.6953i
YS=fftshift(Y)
YS =
   0.0365 - 0.6953i   0.5311 - 0.3254i  -0.3407 + 0.1119i
            1.2964             1.7881             2.1394
   0.0365 + 0.6953i   0.5311 + 0.3254i  -0.3407 - 0.1119i
```

2. 傅立叶积分变换及其反变换

连续和离散的积分变换在应用数学中是重要的工具。拉普拉斯变换作用于连续系统（微分方程），z 变换作用于离值系统（差分方程）。同样傅立叶变换作用于连续函数，而离散傅立叶变换（DFT）作用于有限数据采样。下面介绍傅立叶积分变换及其逆变换。

● 傅立叶变换

在 MATLAB 中，其调用格式为：

◆ *F* = fourier(*f*) 求表达式 *f* 的傅立叶变换。缺省的自变量为 *x*，缺省的返回值是关于 *w* 的函数。如果 *f* = *f*(*w*)，则返回关于 *t* 的函数 *F* = *F*(*t*)；

◆ *F* = fourier(*f*,*v*) 返回函数 *F* 是关于符号表达式对象 *v* 的函数，而不是缺省的 *w*。此时，fourier(*f*,*v*) 等价于 *F*(*v*) = int(*f*(*x*)*exp(-i**v***x*),*x*,-inf,inf)；

◆ *F*=fourier(*f*,*u*,*v*) 傅立叶积分变换最完整的调用格式，是对关于 *u* 的函数 *f* 进行变换，返回函数 *F* 是关于 *v* 的函数，此时 fourier(*f*,*u*,*v*) 等价于 *F*(*v*) = int(*f*(*u*)*exp(-i**v***u*),*u*,-inf,inf)；

【例 6-72】傅立叶变换的应用。

```
syms t v w x
F1=fourier(1/t)
F1 =
```

```
i*pi*(Heaviside(-w)-Heaviside(w))
F2=fourier(exp(-x^2),x,t)
F2 =
pi^(1/2)*exp(-1/4*t^2)
F3=fourier(diff(sym('F(x)')),x,w)
F3 =
i*w*fourier(F(x),x,w)
```

- 傅立叶逆变换

在 MATLAB 中，其调用格式为：

- ◆ f= ifourier(F) 符号表达式对象的傅立叶逆变换。缺省的自变量为 w，缺省返回是关于 x 的函数。如果 $F = F(x)$，则返回关于 t 的函数 $f = f(t)$；
- ◆ f = ifourier(F,u) 返回函数 f 是关于符号表达式对象 u 的函数，而不是缺省的 x 的函数。此时，ifourier(F,u) 等价于 $f(u)$ = 1/(2*pi) * int(F(w)*exp(i*w*u,w,-inf,inf)；
- ◆ f = ifourier(F,v,u) 对关于 v 的函数 F 进行变换，返回关于 u 的函数 f。此时，ifourier(F,v,u) 等价于 $f(u)$ = 1/(2*pi) * int(F(v)*exp(i*v*u,v,-inf,inf)。

【例 6-73】傅立叶逆变换的应用。

```
syms t u w x
f1=ifourier(w*exp(-3*w)*sym('Heaviside(w)'))
f1 =
    1/2/(-3+i*x)^2/pi
f2=ifourier(1/(1 + w^2),u)
f2 =
    1/2*exp(-u)*Heaviside(u)+1/2*exp(u)*Heaviside(-u)
f3=ifourier(v/(1 + w^2),v,u)
f3 =
    -i/(1+w^2)*Dirac(1,u)
f4=ifourier(sym('fourier(f(x),x,w)'),w,x)
f4 =
f(x)
```

6.9 稀疏矩阵

在工程实践中，往往需要解大型的线性方程组。而对于一个用矩阵描写的线性方程来说，n 个未知数的问题就涉及一个 $n \times n$ 的方程组，解这个方程就需要 n^2 个数字的内存和正比于 n^3 的计算时间。这样解一个大型（如上千阶）方程组就很不容易处理（即使现在的计算机技术不断提高），有时对于计算所用的时间要求更为苛刻。但是在大多数情况下，

一个未知数只和数量不多的其他变量有关，即关系矩阵是稀疏矩阵。

在 MATLAB 中，有一种解决该问题的方法，即使用稀疏矩阵。它只存储矩阵的少量非零元素，而不存储那些零元素，也不对它们进行操作，从而节省了内存和时间。稀疏矩阵计算的复杂性和代价仅取决于稀疏矩阵的非零元素数目，而与该矩阵的总元素个数无关。

6.9.1　稀疏矩阵的存储

在 MATLAB 中有两种存储矩阵的方式：一种是全元素（Full）存储，另一种是稀疏（Sparse）存储。全元素存储就是存储矩阵的每一个元素，零值元素和其他元素一样需要相同的存储空间；而稀疏存储仅存储那些非零元素及其下标，对于有较多零元素的大型矩阵，这种存储将显著减少所需的存储空间。稀疏存储有以下特点：

- 只存储矩阵的非零元素，并且存储按列进行；
- 对于每列，用一个实数（或复数）数组记述非零元素值，再用一个整数数组记述相应非零元素的行下标；
- MATLAB 需要三个数组存储实型的稀疏矩阵。如一个 $m\times n$ 的稀疏矩阵，它含有 nz 个非零元素，第一个数组采用浮点形式存储非零元素，该数组的长度为 nz；第二个数组是存储非零元素所对应的行下标，长度也是 nz；第三个数组包含指向每一列开始的指针，它的长度等于 n。这个矩阵的存储包含 nz 个浮点数和 $nz+n$ 个整数。如果稀疏矩阵是用复数矩阵存储，则 MATLAB 还需要第四个数组存储非零复数的虚数部分，它的大小也是 nz。

在 MATLAB 中，实现两种存储方式的转换的命令格式为：

- *SM*=sparse(*A*)　把矩阵的存储方式从任何一种形式（包含有全元素的形式）转变为稀疏矩阵；
- *FM*=sparse(*A*)　把矩阵的存储方式从任何一种形式（包含有稀疏形式）转变为全元素的形式。

【例 6-74】稀疏矩阵和全元素矩阵之间的转化。

```
A=[2 1 0 0 0;0 3 4 0 0;0 0 1 2 0;0 0 0 9 8];              %输入矩阵 A
S=sparse(A)        %把全矩阵 A 转化为稀疏矩阵 S，输出矩阵 S 的非零元素以及对应
                    的行和列
S =
    (1,1)        2
    (1,2)        1
    (2,2)        3
    (2,3)        4
    (3,3)        1
    (3,4)        2
    (4,4)        9
    (4,5)        8
A=full(S)          %把稀疏矩阵 S 转化为全矩阵
```

```
A =
     2     1     0     0     0
     0     3     4     0     0
     0     0     1     2     0
     0     0     0     9     8
```

从矩阵 S 可以看出，在稀疏矩阵中，元素按列存储，这也反映了其内部的数据存储结构。

将全元素矩阵转化为稀疏矩阵是一种建立稀疏矩阵的方法，但并不常用。如果矩阵的阶数足够小，使得矩阵可以采用全元素存储，则将它转化为稀疏矩阵，但不会节省太多的存储空间。

6.9.2 稀疏矩阵的创建

全元素矩阵经过运算后仍然是全元素存储的。因此，如果不经过专门定义和运算，稀疏矩阵将不会自动产生，稀疏矩阵的创建需要用户来决定。但是，一旦某个矩阵以稀疏矩阵方式存储，那么有它参与运算的结果仍以稀疏矩阵方式存储，除非运算本身（如多次加运算）使稀疏性消失。

1. 直接创建稀疏矩阵

在 MATLAB 中，有以下几种方法创建稀疏矩阵：

- 将全元素存储转化为稀疏矩阵存储

这种方法前节已经讲述，此处不再重复。

- 直接创建稀疏矩阵命令 sparse

在 MATLAB 中，直接创建稀疏矩阵的调用格式为：

***SM* = sparse(*i*,*j*,*s*,*m*,*n*,nzmax)**

创建稀疏矩阵最完整的调用格式。

其中：s 是按列排列的所有非零元素构成的向量；i、j 分别是非零元素的行下标和列下标向量。这三个向量的长度相同；m、n 分别是待生成的稀疏矩阵 ***SM*** 的行和列的维数；nzmax 是用来为非零元素指定存储空间的正整数（它一定不小于非零元素总数）。

该调用格式是用[i, j, s]的行创建 $m\times n$ 维稀疏矩阵 ***SM***。

调用格式中的六个输入参数在一定条件下可以缺省。详见 MATLAB 系统的用户指南或联机帮助。

在第六个参数 nzmax 中，如果给定一个较大的数来确定非零元素的最大个数，则以后继续增加非零元素而不需要改变存储要求。

【例 6-75】直接创建稀疏矩阵。

```
SM=sparse([3 1 2 4],[1 2 3 4],[12 3 2 4],4,4,4)
SM =
     (3,1)        12
     (1,2)         3
     (2,3)         2
     (4,4)         4
```

2. 稀疏带状矩阵创建命令 spdiags

在 MATLAB 中，创建稀疏带状矩阵的调用格式为：

A = spdiags(B,d,m,n)

抽取和创建带状、对角稀疏矩阵。

其中：m、n 分别是指定矩阵 A 的行和列；d 是长度为 p 的整数向量，它指定 A 的对角线位置；B 是全元素矩阵（但不必一定），用来给定 A 的对角线位置上的元素，其行维为 min(m,n)，列维是 p。

MATLAB 还提供了其他几种特殊的稀疏矩阵的创建命令，如：稀疏单位阵（Speye），稀疏随机阵（Sprandn）和稀疏对称随机阵（Sprandsym）。

该命令的其他调用形式请参看用户指南或联机帮助。

【例 6-76】创建带状稀疏矩阵。

```
n=5;
e = ones(n,1);
A = spdiags([e -2*e e], -1:1, n, n)        %创建带状稀疏矩阵
A =
    (1,1)        -2
    (2,1)         1
    (1,2)         1
    (2,2)        -2
    (3,2)         1
    (2,3)         1
    (3,3)        -2
    (4,3)         1
    (3,4)         1
    (4,4)        -2
    (5,4)         1
    (4,5)         1
    (5,5)        -2
AA=full(A)                                  %将 A 转化为全元素矩阵 AA
AA =
    -2     1     0     0     0
     1    -2     1     0     0
     0     1    -2     1     0
     0     0     1    -2     1
     0     0     0     1    -2
```

3. 从外部文件输入稀疏矩阵命令 spconvert

MATLAB 允许从外部输入稀疏矩阵。将 load 命令和 spconvert 函数联合使用，可以输入包含一系列下标和非零元素的文本文件。save 命令和 load 命令也可以处理二进制形式存储在 mat 文件中的稀疏矩阵。一个矩阵在 MATLAB 内存空间中以稀疏矩阵形式存在，

在用 save 命令存储成 mat 文件后，用 load 命令调用后仍然以稀疏矩阵的形式存在。

函数 spconvert 命令的调用格式为：

***S* = spconvert(*D*)**

把保存来自外部的数据变量内容转化为稀疏矩阵。

其中：*D* 是来自外部的数据文件，可以是.mat 文件，也可以是.dat 文件；*S* 是指定的稀疏矩阵。

先用命令 load 把数据文件装载于 MATLAB 的内存空间。

D 数组的行维为 *nz*（即非零元素的个数）或 *nz*+1，列维为 3（对实数而言）或 4（对复数而言）。D 数组的每一行（以[*I*，*J*，Sre，Sim]形式）指定一个稀疏矩阵元素。

Load 和 save 命令本身不区分矩阵的存储形式，对任何存储形式，都能正确操作。

【例 6-77】从外部输入稀疏矩阵。

假定已经存在一个文本文件 D.dat，其内容为：

```
8  1  6.00
3  5  7.00
4  9  2.00
9  9     0
```

在 MATLAB 的命令窗口中输入如下命令，即可将 D.dat 调入 MATLAB 中，然后将它转化为稀疏矩阵。

```
load D.dat            %把 D.dat 文件装入 MATLAB 的内存空间
S=spconvert(D)        %把 D 转化为稀疏矩阵
S =
   (8,1)          6
   (3,5)          7
   (4,9)          2
save d S              %将工作空间中的变量 S 存储在文件 d.mat 中
clear                 %清除所有工作空间中的变量
load d S              %将 d.mat 中的变量读入工作空间
S                     %查看变量 S 中是否仍然为稀疏矩阵
S =
   (8,1)          6
   (3,5)          7
   (4,9)          2
```

注意：在外部文本文件 D.dat 中，第一列是一些行下标，第二列是一些列下标，第三列是一些非零元素值（也可以是零值）。

提示：MATLAB 6.0 新提供了一个 Import Wizard 的功能，可以输入外部各种类型的数据文件。读者可查看第 2 章中的相关内容。

6.9.3 稀疏矩阵的查看

MATLAB 提供了下列函数，它们能够对稀疏矩阵进行定量的了解和得到有关的图形

信息。这些函数分别为：

- whos 命令；
- nnz 命令、nonzeros 命令和 nzmax 命令；
- find 函数和稀疏矩阵。

6.9.4 稀疏矩阵的运算

大多数的 MATLAB 标准数学函数，对于稀疏矩阵的运算就像对全元素运算一样。稀疏矩阵的计算复杂性与矩阵中非零元素的个数成正比，计算的复杂性也线性地依赖矩阵行和列的大小，但是与矩阵总的元素个数无关。通常稀疏矩阵运算所需的时间正比于对非零元素进行的算术操作的数目。也就是说，计算时间正比于每秒浮点运算次数。

1. 运算规则

一般说来，MATLAB 系统的各种命令适用于任何一种存储方式的矩阵。全元素矩阵运算总是以全元素形式给出结果。当有稀疏矩阵参与运算时，所得的结果将遵循以下规则：

- 把矩阵变换到标量或定长向量的函数总是给出全元素结果。也就是说，对于一个函数，输入参数是矩阵，而输出的参数是标量或向量，则输出结果采用全元素形式存储。如函数 size、nnz 等。
- 把标量或定长向量变换到矩阵函数（如 diag、eye、ones、randn、zeros）总是给出全元素结果。也就是说，函数输入的参数是标量或向量，输出的参数是矩阵（如提到的几个命令）总是返回全元素形式。能给出稀疏结果的相应命令有 spdiags、speye、sprandn 和 sparse 等。
- 从矩阵到矩阵或向量的“单矩阵”变换函数将以原矩阵（稀疏还是全元素）形式给出结果。也就是说，一元函数的输入是矩阵，输出为矩阵或向量，它将保留操作数的存储特性。若 *S* 是一个稀疏矩阵，则函数 chol(*S*)、diag(*S*)、max(*S*)、sum(*S*)、~*S* 等返回的都是稀疏形式。
- 两个矩阵运算（+，*，\，|）操作后的结果一般是全元素形式，除非参与运算的两个矩阵都稀疏，或操作本身（如.*，&）保留稀疏性。也就是说，如果两个操作数都稀疏，则产生的结果也稀疏；如果两个操作数都是全元素，则产生的结果也是全元素。对于混合操作数，除非运算保留稀疏性，否则将给出全元素结果。
- 参与的矩阵扩展（如[*A B*; *C D*]的子矩阵）中，只要有一个是稀疏的，则所得的结果也稀疏。也就是说，矩阵采用 cat 函数或方括号连接对于混合算子将产生稀疏结果。
- 在矩阵赋值语句中，将以原矩阵形式（稀疏还是全元素）给出结果。也就是说，若矩阵 *S* 稀疏，不管 *i*，*j* 是标量或是向量，在赋值语句右侧的 *T*=*S*(*i*,*j*)将产生稀疏结果；而赋值语句左侧的子矩阵索引 *T*（*i*,*j*）=*S* 中，将产生全元素的结果。

2. 稀疏矩阵运算的常用命令

在 MATLAB 中，提供有针对稀疏矩阵进行运算的函数命令，常用运算命令如表 6-11 所示。

表 6-11 常用稀疏矩阵运算指令

函数名称	功能简介	函数名称	功能简介
issparse	当矩阵稀疏时给出 1，否则为 0	condest	估计 1-范数条件数
nnz	矩阵的非零元素总数	normest	估计 2-范数
nonzeroe	矩阵的非零元素数值	sprank	结构秩
nzmax	指定存放非零元素所需内存	gplot	依图论法则画“（无向）图”
spalloc	为非零元素配置内存	spy	绘出稀疏结构图
sprun	求个非零元素的函数值	etree	消去树
spones	用 1 置换非零元素	etreeplot	画消去树
colmmd	列最小排序	treelayout	展开树、林
colperm	计算列排序置换向量	treeplot	画树图
dmperm	矩阵的 DulTnageMendelsohn 分解	symmd	对称最小排序
randperm	随机置换向量	find	找非零元素的下标
symrcm	反向 Cuthill-McKee 排序		

关于上述函数指令的详细解释请查用户指南和在线帮助。

【例 6-78】全元素矩阵、稀疏矩阵和最小排序稀疏矩阵三角分解所需时间的比较。

```
n=250;                      %给出矩阵的阶数
rand('state',1)             %为重复产生相同的矩阵而设
rand('state',2)
A=sprandsym(n,0.015)+100*speye(n,n);  %建立（100*100）随机正定稀疏矩阵
subplot(1,2,1)
spy(A,'b',10)
title('矩阵 A 的结构图')
subplot(1,2,2)
d=symmmd(A);                %采用最小排序法
spy(A(d,d),'b',10)
title('最小排序法')
B=full(A);                  %给出 A 的全元素形式
%下面比较三个矩阵的 cholesky 分解
format long e
tic,L1=chol(B);t1=toc       %全元素时，分解所用的计算时间
tic,L2=chol(A);t2=toc       %稀疏时，分解所用的计算时间
tic,L3=chol(A(d,d));t3=toc  %最小排序时，分解所用计算时间
t1 =
    1.099999999999994e-001
t2 =
    6.000000000000227e-002
t3 =
    0
```

程序运行结果如图 6-12 所示。

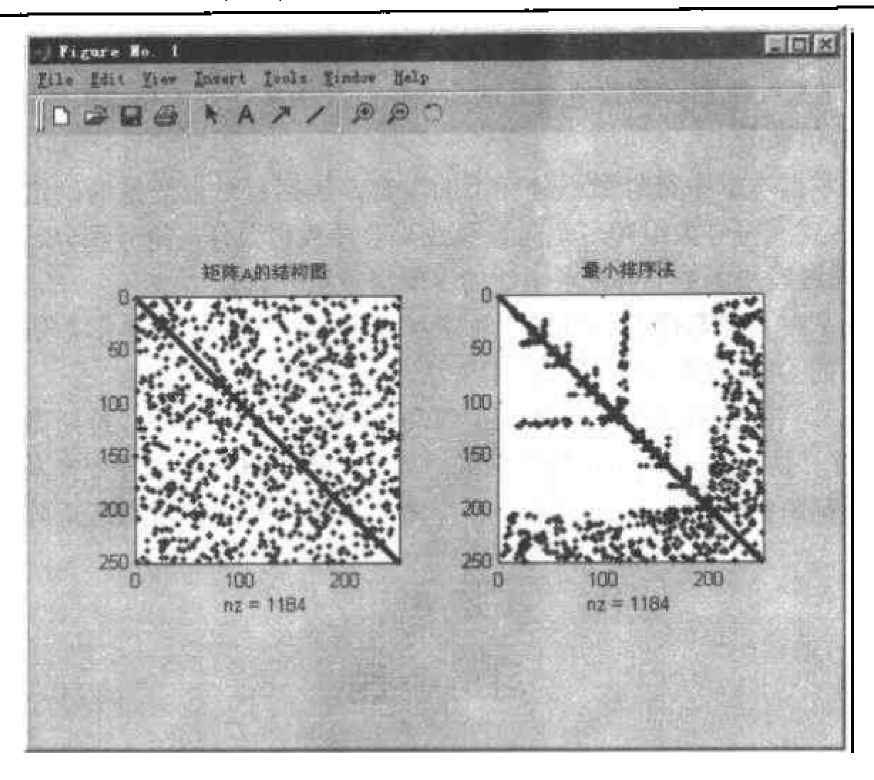


图 6-12　稀疏矩阵结构

在稀疏结构绘图命令 spy 中，第二个参数定义画图颜色，第三个参数使色点较大，便于观察。

指令 symmmd 采用最小排序法，其中心思想是根据消元的每一个阶段和选取可能的主元，获得填充项和运算次数的最小化，因而列高斯消元法能够与最小度算法较好地结合，其运算时间最短。

计算表明：全元素的 cholesky 分解所用时间最多，而经最小度排序后的稀疏矩阵分解所用时间最少。

6.10　小　　结

本章从工程教学的角度，详细并系统地介绍了 MATLAB 在高等数学、线性代数以及符号运算数据处理等方面的应用，这一章是全书最重要、最核心的部分。通过本章的学习，应该重点掌握如下内容：

（1）能对矩阵作多种变换和运算，包括求解矩阵的特征值、特征向量和矩阵的对角化等，熟练掌握各类方程组的多种解法。在解方 的过程中，注意，数组运算和符号运算之间的区别和联系。

（2）了解和掌握多项式的创建和基本运算，熟练掌握多项式的各种化简、提取和替换命令，掌握多项式因式分解和展开。

（3）初步掌握曲线拟合的方法，学会多项式拟合和非线性最小二乘估计。

（4）在插值和样条方面，要掌握一维插值、二维函数插值和样条函数插值的方法。

（5）熟练掌握一维和多重数值积分的命令，以及用多项式求导法求数值微分和用 diff 计算插分法求数值微分。

（6）熟练掌握本章中符号微积分应用的内容，包括符号自变量的确定、求函数的极限、对符号表达式（符号数组和多元向量函数）求导数和微分、符号积分、符号求和等，同时熟练掌握通过调用函数 taylor 求函数的泰勒级数展开式。

（7）熟练掌握各类常微分方程的各种求解方法和函数命令，包括数值解和符号解，了解 MATLAB 的 ODE 文件模板及其使用方法。

（8）熟练掌握数据分析函数的基础运算和有限差分，包括拉普拉斯微分算子、数值梯度、向量运算、协方差矩阵和相关阵；根据需要，熟练掌握傅立叶变换及其逆变换。

（9）掌握稀疏矩阵的各种创建方法和存储，了解并初步掌握稀疏矩阵的查看和运算命令及其用法。

习　题

1．对矩阵 $A=\begin{bmatrix}3 & 1 & 2 & 4\\ 1 & 3 & -2 & 7\\ -2 & 1 & -2 & 4\\ 5 & 6 & 8 & 0\end{bmatrix}$ 作 LU、QR、Schur、奇异值和 Hessenberg 分解，并验证它们的正确性。

2．求下列方程组的解：

① $\begin{cases}2x_1-x_2+3x_3-x_4=3\\ 3x_1+x_2-5x_3+2x_4=0\\ 4x_1-x_2+x_3=3\\ x_1+3x_2-13x_3+3x_4=-6\end{cases}$　② $\begin{cases}2x_1-x_2=3\\ 3x_1+x_2-5x_3=0\\ 4x_1-x_2+x_3=3\\ x_1+3x_2-13x_3=-6\end{cases}$

③ $\begin{cases}2x_1-x_2+3x_3-x_4=3\\ 3x_1+x_2-5x_3+2x_4=0\\ 4x_1-x_2+x_3=3\end{cases}$

3．演示如下输入法的异同，并给出 *a*1-*a*2 的值。

① *a*1=sym([1/3,0.2+sqrt(2),pi])

② *a*2=sym('[1/3,0.2+sqrt(2),pi]')

③ *a*3=sym('[1/3　0.2+sqrt(2)　pi]')

4．求下列矩阵 *A* 的特征值和特征向量，并将矩阵 *A* 转换为符号矩阵，求其特征值和特征向量，比较它们的不同。

A =[-1　1　0
　-4　3　0
　1　0　2]

5．判断下列矩阵是否可以对角化，如果可以，求其逆矩阵和对角阵。

① [1　1　–2
4　0　4
1　–1　4]

② [1　2　3
0　1　0
2　1　2]

6．将下列矩阵进行 QRQ 对角化。



① [0 -1 -1
-1 0 -1
-1 –1 0]

② [3 0 1
0 2 0
1 0 3]

7．求下列矩阵的 Jordan 标准形。

① [3 0 8
3 -1 6
-2 0 5]

② [-4 2 10
-4 3 7
-3 7 1]

8．设 $f(x)=x^5-x^4+2x^3+x^2+3$，$g(x)=x^4+x^2+x-1$，对 $f(x)$ 和 $g(x)$ 进行因式分解。

9．求多项式 $x^4-5x^3-14x^2-10x-3$ 的根。

10．对由余弦函数生成的原始数据点进行一维插值和样条函数插值，并比较它们的异同。

11．用 6 阶多项式拟合正弦函数 sin(*x*)，并利用多项式的求导来求 π 处的一阶和二阶导数。

12．计算下列极限：

① $\lim\limits_{x\to0}\dfrac{1-\cos x}{x^2}$　　② $\lim\limits_{x\to0}\dfrac{1-\cos 2x}{x\sin x}$

13．用 diff 命令求复杂函数 $y=e^{-2x}\cos(3x^{1/2})$的 4 阶导数，并对表达式进行简化.

14．求符号矩阵 *F* 对变量 *x* 的一阶微分、对变量 *a* 的二阶微分.

F=[cos(*a***x*)　sin((*a*+*b*)^2**x*)
-sin(*b***x*)　e^a];

15．建立一个符号表达式 *y*=sin((*a*+*b*)^2**x*+*c*)，分别以变量 *c* 进行积分、以变量 *x* 从 pi/2 到 pi 进行积分，并对 15 题中的符号矩阵 *F* 进行积分。

16．用符号求和函数求级数 $\dfrac{1}{1\cdot2}+\dfrac{1}{2\cdot3}+\cdots+\dfrac{1}{n\cdot(n+1)}$ 的和。

17．将正弦函数 sin(*x*)在 *x*=0 处泰勒展开多项式。

18．计算函数 f=x^2+y^2-x 在区域 *D* 上的积分 $\iint\limits_D(x^2+y^2-x)\mathrm{d}\sigma$，其中 *D* 由直线 *y*=2，*y*=*x* 及 *y*=2*x* 所围成的闭区域。

19．求解下列微分方程：

① $y'=(x+y)(x-y)$

② $xy'=y\text{kig}(y/x)$，$y(10)=1$

③ $y'=-x\sin x/\cos y$，$y(2)=1$

20．求微分方程组 $\begin{cases} f'=f+2g \\ g'=f+4 \end{cases}$ 的解。

21．求微分方程组 $\frac{\mathrm{d}f}{\mathrm{d}t}=f+g,\frac{\mathrm{d}g}{\mathrm{d}t}=-f+g,$，当初始条件为 $f(0)=2$，$g(2)=5$ 时的解。

22．建立一个 5×5 的随机矩阵 A，求其各列元素的最大值。并求各列元素的累积和。

23．对向量 a=(3 4 5)和向量 b=(5 7 8)求点积和叉积的计算。

24．建立两个方阵 A 和 B，并计算方阵 A 的协方差矩阵以及方阵 A 和 B 相关阵。

25．对第 24 题中的矩阵 A 进行快速傅立叶变换。

26．求 $\begin{bmatrix} \delta(t-a) & u(t-b) \\ e^{-at}\sin bt & t^2\cos 3t \end{bmatrix}$ 的 Laplace 变换。

27．验证积分 $\int_{-\tau/2}^{\tau/2} Ae^{-i\omega t}\,\mathrm{d}t = A\tau\cdot\frac{\sin\frac{\omega\tau}{2}}{\frac{\omega\tau}{2}}$ 的正确性。

28．求 $f=\begin{bmatrix} x_1 e^{x_2} \\ x_2 \\ \cos(x_1)\sin(x_2) \end{bmatrix}$ 的 jacobian 矩阵。

29．求积分 $\int_1^2\int_{\sqrt{x}}^{x^2}\int_{\sqrt{xy}}^{x^2 y}(x^2+y^2+z^2)\,\mathrm{d}z\,\mathrm{d}y\,\mathrm{d}x$ 。注意，内积分上下限都是函数。

30．调用 MAPLE 函数求递推方程 $f(n)=-3f(n-1)-2f(n-2)$ 的通解。

31．用两种不同方式创建三对角稀疏矩阵。

第7章　句柄图形和GUI程序设计

以图形界面为主的人机交互方式，因其操作和控制的方便、灵活而逐渐成为现代应用程序的主要交互方式。MATLAB 6.0 对图形界面做了重大的改进和调整后，在图形用户界面方面具有更强大的功能。本章主要介绍了 MATLAB 6.0 的句柄图形内容和图形界面，通过大量实例讲述了图形用户界面的设计和制作。

所谓用户界面（User Interface）就是用户与计算机之间交互通信联系的平台。现在的应用程序出现了多种形式的人机交互方式，逐渐从原来的命令行交互方式转变为以图形界面为主的交互方式，由于图形界面的操作和控制方便、灵活，使得图形界面逐渐在人机交互方式中占据了主导地位。

图形用户界面（Graphic User Interface）是指包含图形对象的用户界面，它包含两类基本的图形对象：一是用户界面控件对象（Uicontrol），简称为控件对象；一是用户界面菜单对象（Uimenu），简称为菜单对象。MATLAB 提供了在其应用程序中加入 GUI 的功能，它是 MATLAB 图形句柄系统的子系统，因此，本章将先简要介绍 MATLAB 的句柄图形，再详细阐述关于 MATLAB 的 GUI 特性。

7.1　句柄图形

句柄图形（Handle Graphics）是一种面向对象（Object-Oriented）的绘图系统概念，它提供了创建计算机图形所必须的各种软件，所支持的指令可以直接创建线、文字、面以及图形用户界面，它实质上属于 MATLAB 可视化功能的内核部分。

7.1.1　图形对象、对象句柄和句柄图形的结构层次

句柄图形是基于这样的概念：一接图的每一个组成部分都是一个对象（Object），每一个对象有一系列句柄（Handle）和它相关，每一对象都有按需要可以改变的属性。一个对象可以定义为一组紧密相关、形成为一个整体的数据结构或函数。在 MATLAB 中，图形对象是一幅图中很独特的成分，可以被单独地操作。

MATLAB 使用于数据可视和界面制作的基本绘图要素成为句柄图形对象（Handle Graphics Object）。由图形命令产生的每一件东西都是图形对象，它们包接图形窗口或者图形，还有坐标系、线条、曲面以及文本等。这些对象接父对象和子对象组成层次结构。构成 MATLAB 图形体系的 13 个图形对象的层次关系如图 7-1 所示，每个图形对象可以被独立地操作。MATLAB 生成的每个具体图形，都由若干个不同对象构成。每个具体图形不必包含全部对象，但是每个图形必须具备根屏幕（Root）和图形窗（Figure）。

每个父对象可以包含一个或多个子对象。例如在图 7-1 中，根屏幕可以包含一个或多

个图形窗口，一个图形窗口可以包含一组或多组坐标轴。除了控件对象（Uicontrol）和菜单对象（Uimenu）外，所有的对象都是坐标轴的子对象，并且在这些坐标轴上显示。所有创建对象的函数当父对象不存在时，都会创建它们。例如，如果没有图形窗口，plot(*x*,*y*)函数将会用缺省属性创建一个新的图形窗口和一组坐标轴，然后在这组坐标轴内画线。

句柄（Handle）实际上是分配给每一个对象的数字标识（Identifier），每次创建一个对象时，就为它建立一个惟一的句柄。句柄是存取图形对象的惟一规范识别符，不同对象的句柄不可能重复和混淆。每台计算机的根对象只有一个根屏幕，其句柄总是数字 0；而简称为图的图形窗口（Figure Window）句柄总是正整数，它用来标识图形窗的序号。除根对象和图对象外，其余对象的句柄则是双精度浮点数。

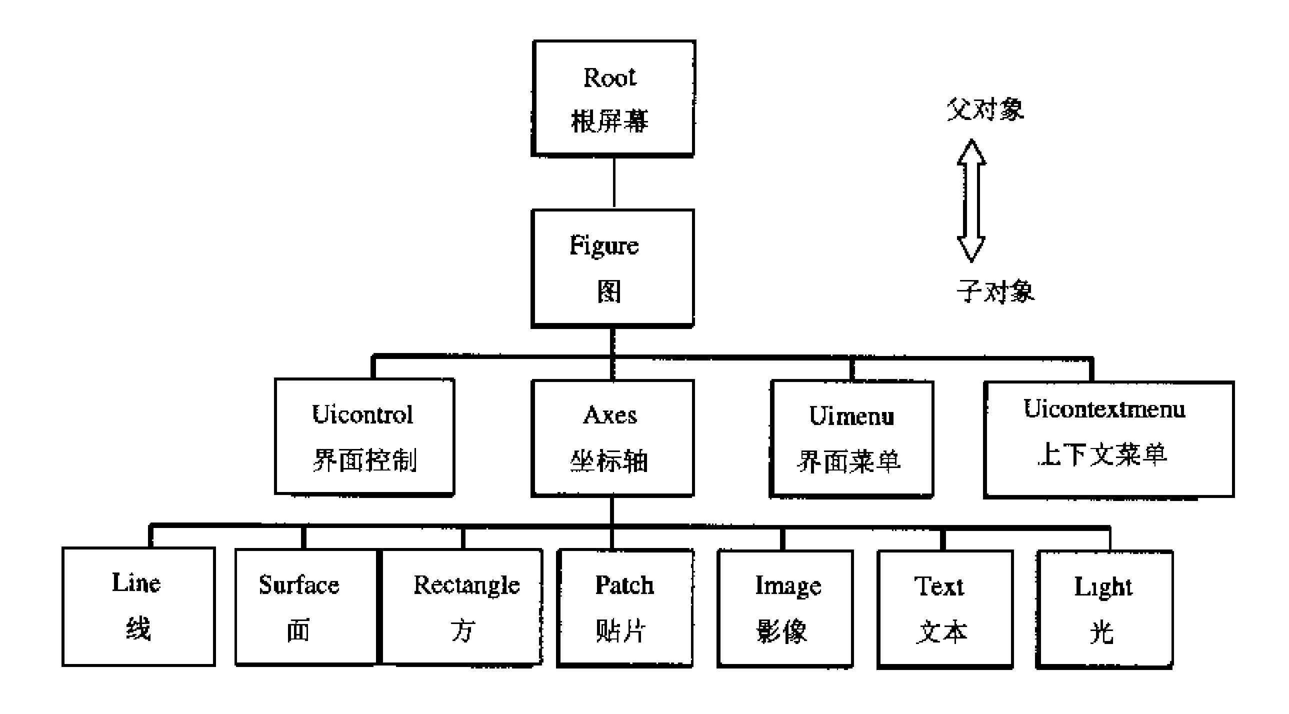


图 7-1　句柄图形体系的对象层次结构图

注意：对于根屏幕、图对象来说，数字可以直接作为调用对象的句柄，但不可企图通过直接输入浮点数，作为其他对象的句柄，这些对象的句柄只能由相关指令运作而得。

在 MATLAB 中，除了根对象 Root 外，所有的对象都由与之相对应的内置函数（Build-in Function）创建，每个函数在创建对象的同时，还可以返回该对象的句柄 *h*。此类函数的名称和调用格式如表 7-1 所示。

表 7-1　　创建图形对象的底层函数

函数名称	功能	调用格式
axes	创建轴	*h*=axes('position',[left, bottom, width, height])
figure	创建图形窗口	*h*=figure(*n*)
image	创建影像	*h*=image(*x*)
line	创建线	*h*=line(*x*,*y*,*z*)
patch	创建贴片（填充多边形）	*h*=patch(*x*,*y*,*z*,*c*)
surface	创建面	*h*=surface(*x*,*y*,*z*,*c*)

续表

函数名称	功能	调用格式
rectangle	创建方形（注意还包括其中的填充）	*h*=rectangle('position',[*x*,*y*,*w*,*h*],'cuivature',[*xc*,*yc*])
text	创建文本	*h*=text(*x*,*y*,'string')
light	创建光	*h*=light('position',X)
uicontrol	创建用户界面控制对象	*h*=uicontrol('property',value)
uimenu	创建用户界面菜单对象	*h*=uimenu('property',value)

关于调用格式中各参数未能描述详尽处，可查看 MATLAB 的帮助文件。需要说明的是，每个底层函数只能创建十个图形对象中的一个，并将它们置于适当的父对象中．例如，line 函数的运作将在当前轴上利用缺省属性值画线，如果在此函数执行前，“轴”和“窗”不存在，MATLAB 将会自动创建它们；如果函数执行前“轴”和“窗”已经存在，那么该“线”将被画在当前“轴上”，并且不影响该轴上已有的其他对象（但这与高层作图不同）。这个特点非常重要，尤其是当图形仅有某一部分需要改变时，读者应用心掌握。

对于某些图形对象，MATLAB 可以用相应函数来获取它们的句柄，直接调用即可返回对象的句柄值，同时这些函数还可作为参数被其他函数调用，如 set 和 get 等对图形句柄操作的函数。此类函数如表 7-2 所示。

表 7-2　获取图形对象句柄的函数

函数	含义和功能
gcf	获取当前图形窗口的句柄
gca	获取当前坐标轴的句柄
gco	获取在当前图形窗口中“被鼠标最近单击”的当前对象的句柄
gcbo	获取当前正在执行调用对象的句柄
gcbf	获取包括正在执行调用对象图形的句柄

在获取图形句柄后，即可对图形对象进行各种操作，其中包括用 delete 命令删除图形对象，如 delete(gca)将删除当前轴和它所有的子对象。

用函数 findobj 可以快速地遍历对象层并获取指定属性的对象句柄，该函数的调用方式如下：

- *h*=findobj('propertyname',propertyvalue,…)　在所有的对象层中查找符合指定属性值的对象，返回句柄值给变量 *h*；
- *h*=findobj(ObjectHandle ,'propertyname',propertyvalue,…)　把查找的范围限制在句柄“ObjectHandle”指定的对象及其子对象中；
- *h*=findobj(ObjectHandles,'flat', 'propertyname',propertyvalue,…)　把查找的范围限制在句柄“ObjectHandle”指定的对象中，但不包括在其子对象中；
- *h*=findobj　返回根对象和所有子对象的句柄值；
- *h*=flndobj(ObjectHandles)　返回“ObjectHandle”指定的对象和其所有子对象的句柄值。

【例 7-1】创建一个图形对象，并寻求图形对象的句柄值。

```
mesh(peaks(30))                    %创建山峰的网格图
text(30,20,2,'\leftarrowpeak')     %给图形对象加上文本，程序执行结果如图 7-2 所
```

```
                            示，此时图形对象中包括坐标轴、线条、文本和标注
h=findobj(gcf)              %求当前图形窗口的句柄
h =                         %返回句柄值赋予变量 h
     1.0000
    72.0009
    74.0010
    73.0039
```

程序的运行结果如图 7-2 所示。

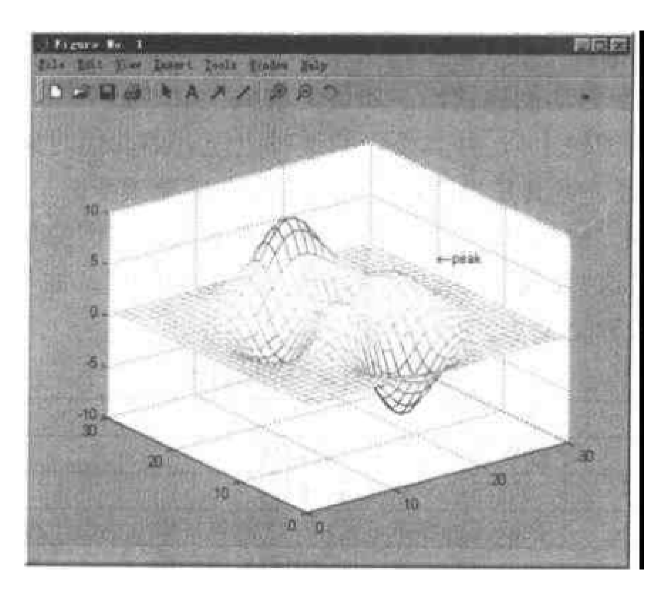


图 7-2 创建的一个图形对象

由程序执行结果可以看出，结果返回了四个句柄值。其中，*h*(1)=1 为图形对象（Figure）的句柄；*h*(2)=72.009 为图的下一级子对象坐标轴的句柄；*h*(3)=74.0010 为坐标轴的下一级子对象线条的句柄；*h*(4)=73.0039 为坐标轴的下一级子对象文本的句柄。可见句柄中的元素排列顺序由各个对象在整个对象层次结构图中的位置决定。

在 MATLAB 中，可以通过函数 copyobj 把对象从一个父对象中复制到另一个父对象中。新的对象和原对象只是句柄和属性“parent”的值不同，其他属性均不变。该函数既可以把几个对象复制到同一个新的父对象中，也可以把一个对象同时复制到几个不同的父对象中，但必须保持正确的“父子关系”。如果复制的对象中含有子对象，复制时将把所有的子对象一并复制。函数的调用格式包括以下几种：

- *C*=copyobj(*H*,*P*) 参数 *H* 和 *P* 都是向量，以向量的元素为对象的句柄，向量 *H* 中每一个句柄对应的图形对象都将被复制到向量 *P* 中相应句柄对应的图形元素之下，分别成为这些父对象的子对象。而这些新对象的句柄将会赋值给向量 *C* 中的相应元素；*H* 和 *P* 的长度必须相等，而且其中的每个对应元素“*H*(*i*)”和“*P*(*i*)”必须具有合法的“父子关系”。
- *C*=copyobj(*H*,*p*) 参数 *H* 是向量，参数 *p* 是标量。向量 *H* 中的每一个句柄对应的图形对象都将被复制到句柄 *p* 对应的图形对象之下，同时成为这个父对象的子对

象。而这些新子对象的句柄将会赋给向量 C 中的相应元素。

- C=copyobj(h,P)　参数 h 是标量，参数 P 是向量。句柄 h 对应的图形对象将被复制到向量 P 中的每个句柄对应的图形对象之下，分别成为这些父对象的子对象。而这些新子对象的句柄将会赋给向量 C 中的对应元素。

【例 7-2】结合例 7-1 说明函数 copyobj 的用法。

```
mesh(peaks(30));
text(30,20,2,'\leftarrowpeak');
hh=findobj(gcf)
```

在原程序之后加入如下内容：

```
h=findobj('string','\leftarrowpeak')      %将文字对象的句柄值返回给变量 h
figure                                     %重新建立一个图形对象
mesh(peaks(20));                           %在 20*20 的区域创建图形内容
hh =                                       %程序输出结果
    1.0000
   72.0012
   74.0012
   73.0037
h =
    74.0012
```

使用 copyobj 命令进行对象的复制，在上述程序后加上如下内容：

ha=copyobj(h,gca)　%将例 7-1 中的文本对象即变量 h 代表的句柄所指的对象复制到图 7-3 中

```
ha =                %程序输出结果
    148.0013
```

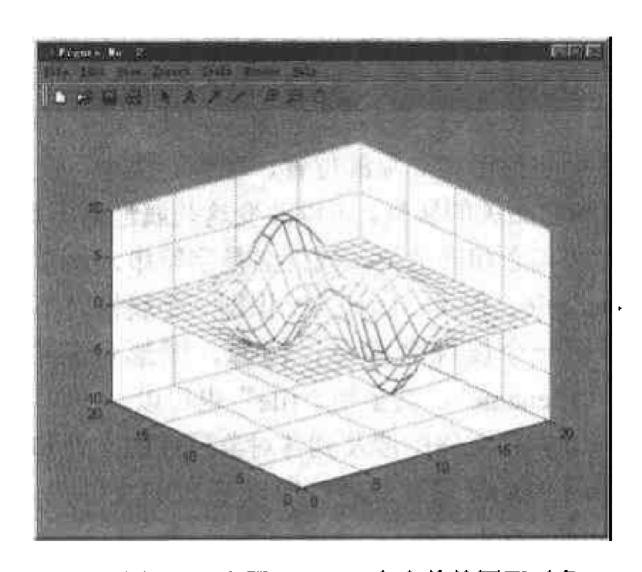


图 7-3　应用 copyobj 命令前的图形对象

程序执行结果如图 7-4 所示。命令中的句柄变量 h 即是文本对象“peak”的句柄值，

而 gca 用来在目的图形对象中获取文本的父对象——坐标轴的句柄值。

同时可以通过本例看出，使用 figure 命令另外建立了一个新的图形窗口，在新图形窗口的标题栏中显示了此图形对象的句柄为“2”。

注意：必须保证要获取句柄的对象或其父对象为当前对象。

当需要进行对象的删除时，可以调用函数 delete 来完成这一操作，调用格式为：

delete(*h*)

使用该命令可以删除句柄 *h* 所指的对象及其所有的子对象。由于该命令不提供确认直接执行，用户在使用时应谨慎进行。

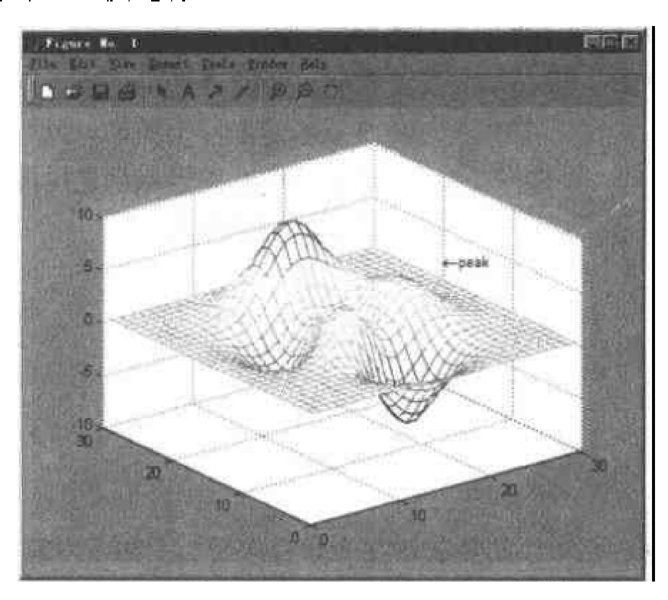


图 7-4　应用 copyobj 命令后的图形对象

7.1.2　图形对象属性的获取及其设置

1. 对象的属性

所有的对象都有属性（Property），用户可通过设定这些属性来定义和修改对象的特征。尽管许多属性所有对象都有，但与每一种对象类型（如轴、线、面）相关的属性列表却是独一无二的。对象的属性可包括诸如对象的位置、颜色、类型、父对象、子对象及其他内容。每一个不同对象都有和它相关的属性，可以改变这些属性而不影响同类型的其他对象。

对象属性包括属性名和与之相应的值，属性名是字符串，通常按混合格式显示，每个词的首字母大写，如 LineStyle。但是 MATLAB 识别一个属性时不区分大小写，只要用足够多的字符来惟一地辨识一个属性名即可，例如，图形对象中的文件名属性一般用“FileName”，也可以是“filename”，甚至用“file”也可以。

下面给出 MATLAB 6.0 帮助文件中涉及到各对象属性的文件和查阅方法。

- 使用 Adibe Acribat Reader 阅读或者打印以下 PDF 文件：
  help\pdf_doc\matlab\graphg.pdf；
  help\pdf_doc\matlab\ref\refbook2.pdf
- 使用命令 help 查看有关对象的资料，如在命令窗口中键入 help figure，可以直接获得有关图形对象的属性描述；

- 在 MATLAB 6.0 的帮助窗口中，键入关键字（如 Graphic Property）搜索、查询关于对象属性的文件；
- 利用 get 和 set 命令在 MATLAB 的命令窗口中直接查询对象的属性。

在本章的学习过程中，当遇到重要的属性时，将给出详细的说明。

2. get 函数和 set 函数

当建立一个对象时，该对象有一组默认属性值，属性值可以通过两种方式来改变。一是用{属性名，属性值}对建立对象生成函数，该方法较为简单；二是通过 get 和 set 函数来获得改变句柄图形对象的属性值。本小节将着重介绍这两个函数的含义和用法。

- get 函数用于获取指定对象的属性（Property），它有如下几种用法：

◆ *v*=get(*H*,'PropertyName') 返回句柄为 *H* 的对象中名为“PropertyName”属性的值。如果 *H* 是向量，函数 get 将同时返回向量 *H* 中每个句柄对应图形对象的指定属性值。

get(H) displays all property names and their current values for
the graphics object with handle H.

◆ get(*h*) 这里 *h* 是标量，返回句柄为 *h* 的对象的所有属性名及其当前取值。

◆ *v*=get(*h*) 这里 *h* 是标量，返回一个结构，结构的每个域名就是句柄为 *h* 的对象的所有属性名，每个域又包括属性的值。

◆ *v*=get(0, 'factory<objecttype><propertyname>') 对于所有类型的对象，返回其所有可以由用户设定缺省值的属性的“出厂值”（即未经过任何用户改动的最初缺省值，尖括号内的内容表示可选。

◆ *v*=get(*h*, 'default<ObjectType><PropertyName>') 返回缺省的属性值。此处句柄 *h* 必须是标量，尖括号内的内容表示可选，如果这两个内容都不选，那么该函数返回句柄为 *h* 的对象所有属性的缺省值。缺省值不能从对象的子对象或对象本身来查询，而只能从对象的父对象来查询。

- set 函数用来设置对象属性值，其调用方法如下：

◆ set(*H*,'PropertyName',PropertyValue) 把句柄为 *H* 的对象中名为 PropertyName 的属性值设置为“PropertyValue”。*H* 可以为向量，此种情况下，set 函数为所有对象设置属性值。

◆ set(*h*,*a*) 此处 *a* 是结构，其域名就是对象的属性名，属性值包括在域中，set 函数把属性值赋给和域名相同的属性。句柄 *h* 为标量。

◆ set(*H*,*PN*,*PV*) 参数 *PN* 是$1\times n$维的数组，其中的元素为需要设置的属性名，*PV* 中的元素是要设置的属性值。

注意：如果向量 *H* 的长度是 m，那么 PV 的维数应该是 $m\times n$。

◆ set(*H*,'PropertyName1',PropertyValue1,'PropertyName2',PropertyValue2,...) 用一条语句同时设置多个属性值。

◆ *A*=set(*H*, 'PropertyName') 或 set(*H*, 'PropertyName') 返回或显示句柄为 *H* 的对象的指定属性值。

◆ *A*=set(*H*)或 set(*H*) 返回或显示句柄为 *H* 的对象的所有属性和可能的取值。

注意：函数 set 和函数 get 返回不同的属性列表。set 函数只列出可以用 set 命令改变

的属性，这一类的属性只可读，但不能改变，称为只读属性；而 get 函数可以列出所有对象的属性。

【例 7-3】创建一个图形对象，练习和体会 set 和 get 函数的用法。

```
x=0:0.1:2*pi;
y1=sin(x);                    %创建一个正弦函数
H1_sin=plot(x,y1,'*')         %画图并返回图像的句柄值
set(H1_sin,'color',[1,0.5,0],'Linewidth',1)    %调用函数 set 设置正弦曲线的颜色和线宽
y2=cos(x);
hold on
H1_cos=plot(x,y2)
set(H1_cos,'color',[0.25 0 1])
title('正弦和余弦函数曲线','fontsize',16,'color','blue')
H_sin_color=get(H1_sin,'color')   %调用 get 函数返回具有句柄 H1_sin 的对象的颜色
```

程序执行结果如下，并产生如图 7-5 所示图形对象。

```
H1_sin =
        78.0005
H1_cos =
        79.0001
H_sin_color =
            1.0000    0.5000         0
```

使用函数 set 和 get 来设置对象的属性非常灵活方便，并且使用简单，功能强大，因此这两个函数经常使用，读者可以在应用中细心体会它们的使用技巧。

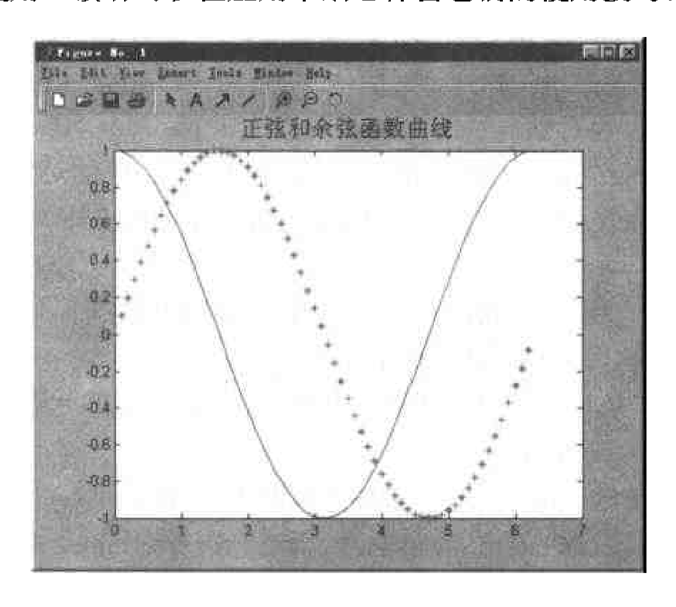


图 7-5 用 set 命令设置后的图形

7.1.3 图形对象的属性编辑器

MATLAB 6.0 提供了专门用于对象属性编辑的属性编辑器（Property Editor）对话框，

本节将对此进行详细说明。

1. 图形窗口的交互操作方式

从 MATLAB 5.3 版本开始，新增加了在图形窗口中直接进行人机对话的交互操作方式。MATLAB 6.0 对此项功能做了进一步的改进工作，专门设计了属性编辑器对话框，在同一个窗口中设置多个标签页，可同时完成多个图形对象的属性编辑。图 7-4 便是一个典型的图形窗口。可以通过菜单或快捷工具栏的按钮来对图形对象进行编辑，单击按钮■，在需要编辑的对象上双击鼠标左键，即可弹出如图 7-6 所示属性编辑器对话框（本例单击坐标轴，出现“Property Editor-Axes”对话框），在此对话框中对该对象进行操作。

- 编辑坐标轴属性对话框

最顶端的空白栏显示此时可编辑的对象名称，下面有多个标签页，分别对应设置坐标轴对象的比例、风格、标注、类型、光源、视点和当前信息。

编辑坐标轴属性对话框如图 7-6 所示，在此对话框中的各个标签页的标题含义如下：

- ◆ Scale　坐标比例；
- ◆ Style　坐标轴的风格；
- ◆ Label　对坐标轴的标注；
- ◆ Aspect　坐标轴的类型；
- ◆ Light　设置光源；
- ◆ View point　设置视点；
- ◆ Info　坐标轴对象的当前信息，是可见还是不可见等。

此处设置坐标轴的属性和使用相关命令进行设置的效果相同。

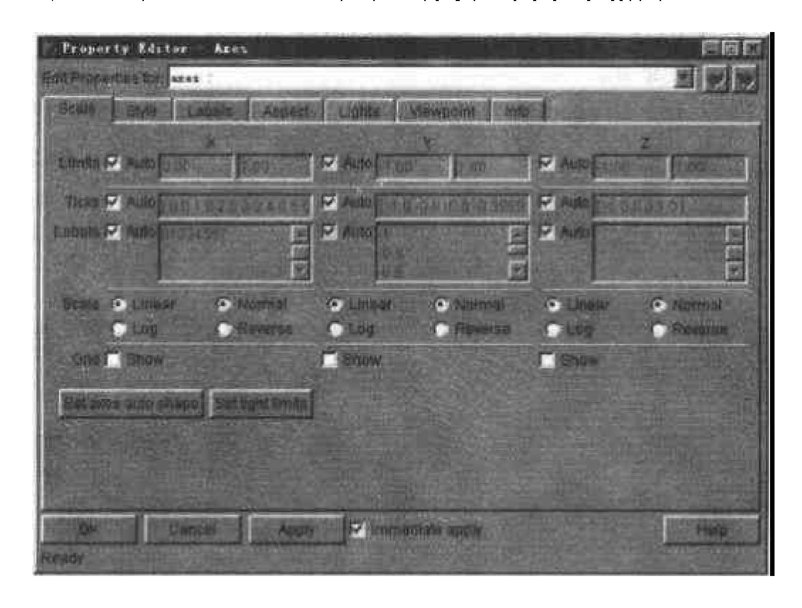


图 7-6　编辑坐标轴属性对话框

图 7-6 中，顶端的空白栏为一下拉式菜单，可显示根屏幕对象的所有子对象信息（如本例中的图形对象包含图形对象 1、坐标轴和两个线条，共 4 个子对象），用户可通过选择相应的子对象来对其进行属性编辑。选择线条 line，将弹出如图 7-7 所示的编辑线条属性对话框。

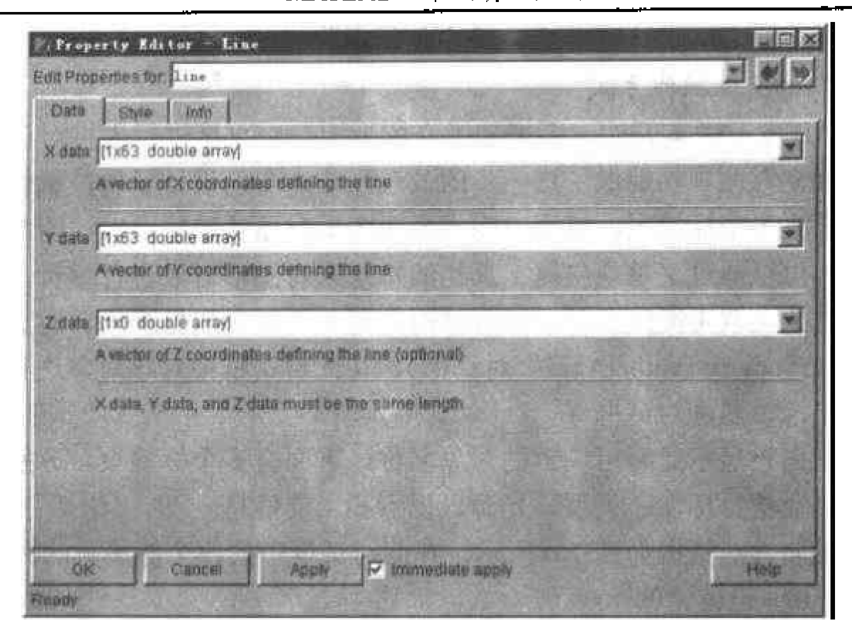


图 7-7 编辑线条属性对话框

- 编辑线条属性对话框

在此对话框中同样含有多个标签页，各个标签页的含义如下：

  - ◆ **Data** 分别定义线条对应的 x 轴向、y 轴向和 z 轴向的向量值。**Line Width**：线条宽度。
  - ◆ **Style** 类型。分别定义线型、线宽、颜色以及是否添加标记符号等。
  - ◆ **Info** 坐标轴对象的当前信息，是否可见等。

- 编辑文本属性对话框

用鼠标左键单击文本，可选中文本对象，单击鼠标右键，将出现如图 7-8 所示下拉式菜单，用户可自行选择以进行所需的操作，通用的编辑字体属性对话框如图 7-9 所示。

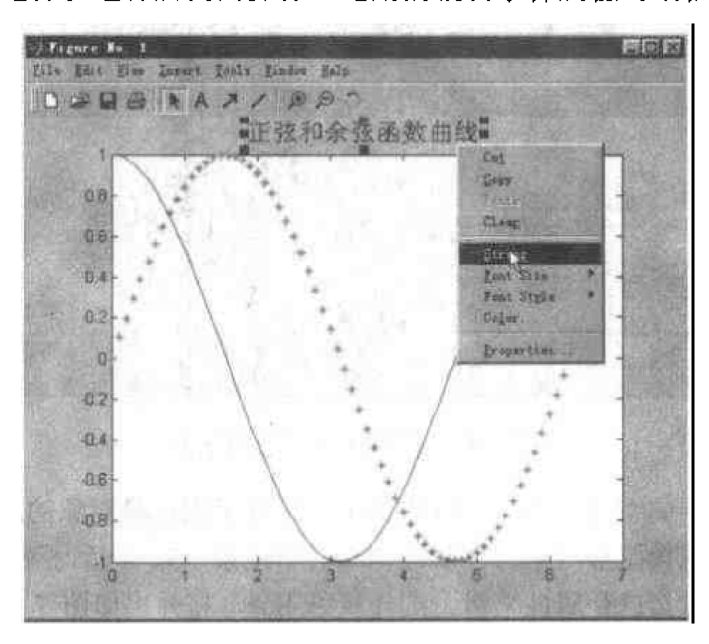


图 7-8 文本对象被选中的状态和弹出式菜单



图 7-9　编辑字体属性对话框

2. 图形属性编辑器

采用图形窗口的交互方式只能对图形窗口中的坐标轴、线条和文本这三种对象的几个基本属性进行交互式编辑，使用虽然简便，但只能对几个基本的属性进行编辑，具有很大的局限性。因此，MATLAB 提供了一个功能强大的属性检查器（Property Inspector）。

属性检查器的部分功能相当于 MATLAB 5.3 版本的图形属性编辑器，它可以对所有图形对象的所有属性进行交互式编辑，实现 get 函数和 set 函数的所有功能。

在命令窗口中直接键入命令 inspect，将弹出如图 7-10 所示属性检查器。

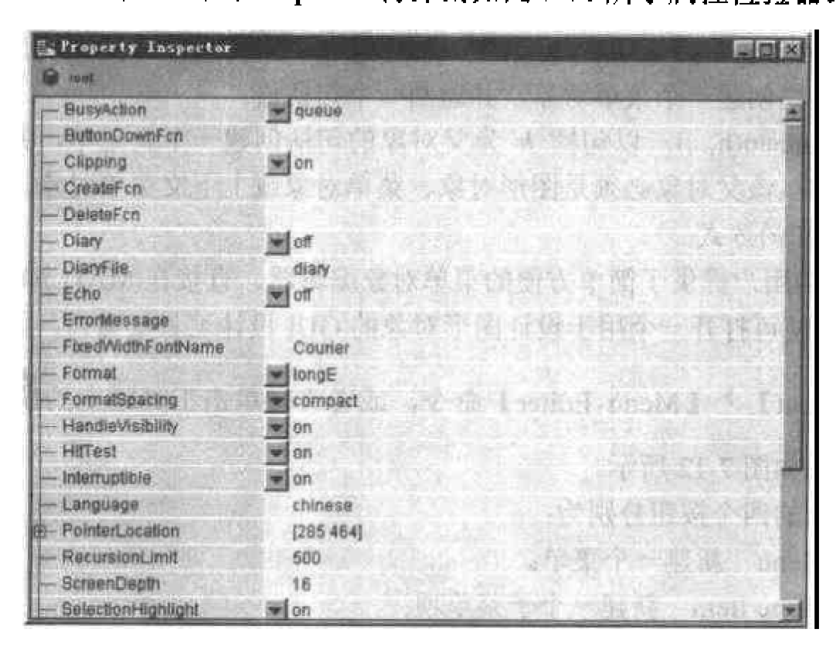


图 7-10　图形属性检查器

最上面的按钮表示所显示属性所属的对象名称（本例为 figure），编辑器下面为属性内容，其中，左侧为对象的所有属性名称，右侧为当前该属性的取值，用户可在此栏对属性的值进行修改和添加。属性检查器显示了当前对象的所有属性值，各种对象按层次排列，

双击左侧有“+”的对象，该对象的下一层即被展开。用鼠标左键单击要编辑的对象，该对象即被选中，同时在图形窗口的显示成为编辑状态。

7.2 用户界面菜单对象和上下文菜单

菜单对象（Uimenu）（有时也称下拉式菜单对象）能够使用户在运行应用程序时，从一批功能选择项中浏览和选择某项功能。在 MATLAB 每个图形窗口上都有一个主菜单栏，所有的主菜单都列在菜单栏上。菜单的标题或名字简单地描述了该菜单的功能。由图 7-1 所示层次结构图中可知，uimenu 对象以 figure 图形窗对象为父对象，和轴（Axis）、用户界面控制对象（Uicontrol）等是同等的兄弟对象。

7.2.1 菜单对象的创建

创建菜单对象有两种方式：一是基于函数命令行的编程方式，二是基于 GUI 的方式。

1. 基于函数命令行的编程方式

自制用户菜单对象，通过函数 uimenu 创建，可用于创建菜单对象（或称子菜单对象）和命令对象（或称菜单项）。菜单对象是指自身包含有下一级命令，功能是打开它的子项；而命令对象是指本身不再具有子菜单的功能选项，只对应于某种功能操作。二者区别在于：命令对象的 Children 属性值为空矩阵，而菜单对象的 Children 属性值为图形窗口句柄值或另一个菜单对象的句柄值。

uimenu 函数的调用格式为：

- uimenu('PropertyName1',value1,'PropertyName2',value2,...)　在当前图形窗口上部的菜单栏创建一个菜单对象，并返回一个句柄值；
- Hm=uimenu(h,...)　以句柄 h 为父对象的句柄创建一个新菜单，并返回该菜单的句柄值：该父对象必须是图形对象、菜单对象或上下文菜单对象。

2. 基于 GUI 的方式

MATLAB 为用户提供了简单方便的菜单对象编辑器。直接在 MATLAB 命令窗口中键入 guide 命令，便可打开一个用于设计图形对象的 GUI 设计工具集窗口，如图 7-11 所示。

单击【Layout】▶【Menu Editer】命令，或者直接单击工具栏上的按钮，即可打开菜单编辑器，如图 7-12 所示。

图中最上面的四个按钮分别为：

- New Menu　新建一个菜单；
- New Menu Item　新建一个子菜单项；
- New Contextmenu　新建一个上下文菜单；
- Delete Selected Item　删除选定的菜单栏。

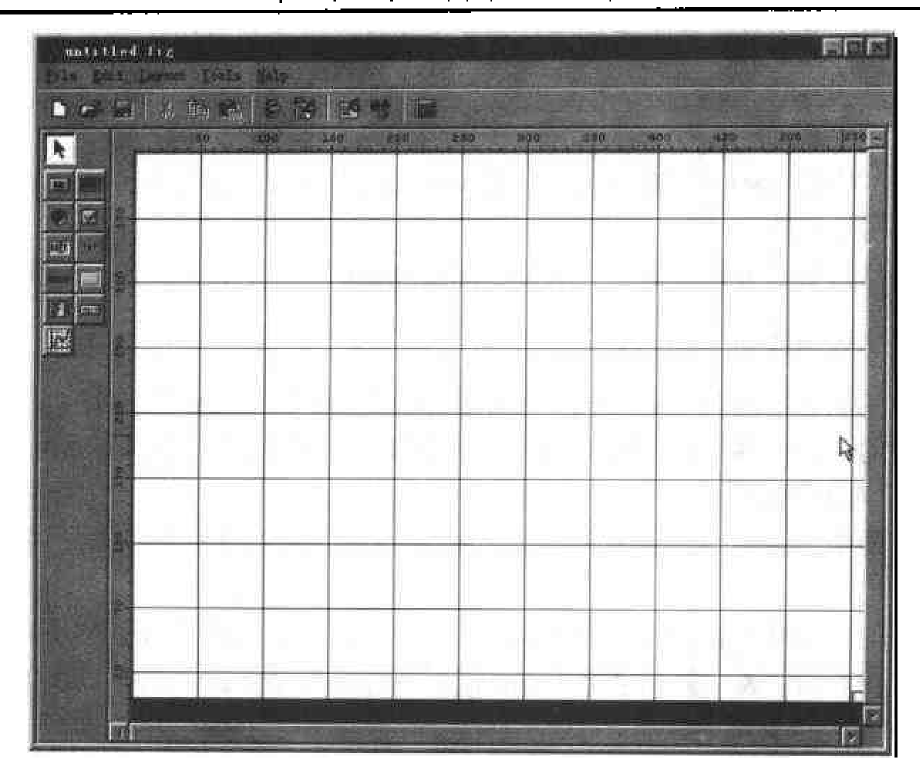


图 7-11　GUI 设计工具集

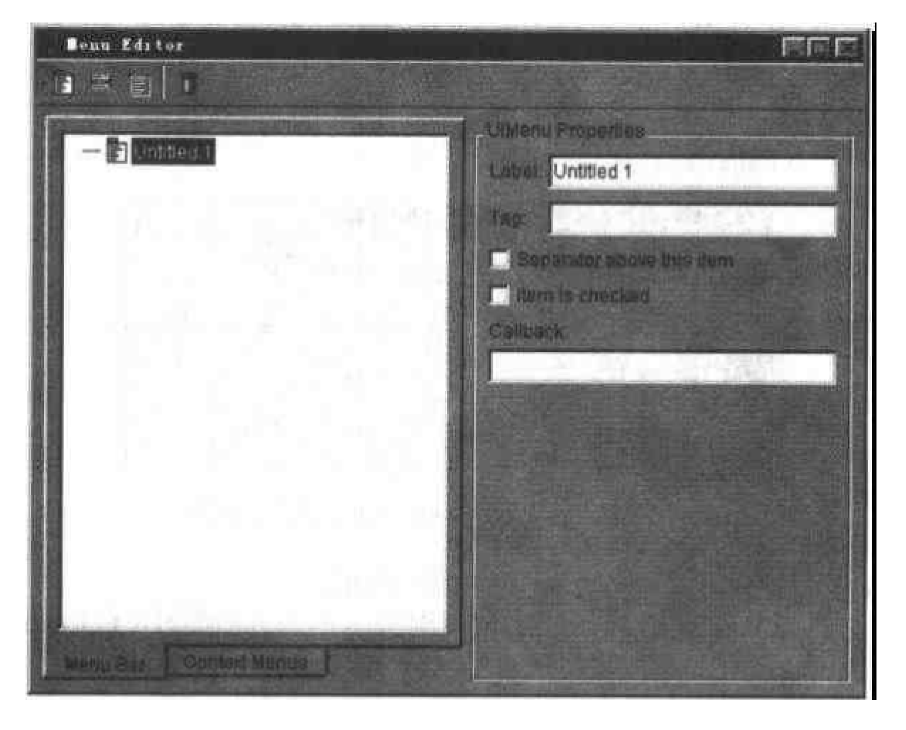


图 7-12　菜单编辑器

左边窗口给出当前图形中的菜单项列表，右边给出相应的对象属性，分别是菜单项的标注（Label）、标记（Tag）和回调程序（Callback）的内容。用户可直接对之进行修改和添加。在菜单项的标注字符串中可以定义相应的快捷键，从而能够使用组合键 Alt+字符来打开相应的菜单项。方法是在指定标注字符串时，在该字符的前面加上符号“&”。

注意：在同一层次的菜单中不能有相同的快捷键。使用菜单编辑器，只能看到菜单的 Label、Tag 和 Callback 这三个属性，如果要对别的属性进行修改，可以使用 Callback 编辑器或者属性检查器。

回调程序（Callback）的内容可以是 MATLAB 的函数、命令和可执行的表达式，也可调用自己编写的函数。

菜单编辑器的第二个标签页尾“New Context Menu”用于新建一个上下文菜单。

7.2.2 菜单对象的属性

菜单对象的属性可分为公共属性、基本控制属性和 Callback 管理属性三部分，关于属性及其取值的详细内容参见 MATLAB 的帮助文件，这里给出一些重要属性的设置方法。

- Label 和 Callback

这是菜单对象的基本属性，编写一个具有基本功能的菜单对象必须要设置 Label 和 Callback 属性。

Label 是在菜单项上显示的标注文本，在文本中同时可以设置该菜单项的快捷键。设置的方法为：把符号“&”放在字符串中，用于快捷键的字符前，执行该菜单项时就可以用快捷键 Alt+该字符来完成。

例如，用如下命令在当前图形窗口的菜单条上建立一个菜单并设置其快捷键：

```
h1=uimenu('Label','&Draw');
```

此时，便在菜单栏上建立了一个新的菜单项【Draw】，快捷键为 Alt+d 或者 Alt+D。如果是'D&raw'，则快捷键为 Alt+r。

属性 Callback 用来设置菜单项回调程序。如下命令将在菜单【Draw】下建立子菜单，并给出回调程序。此时的图形窗口如图 7-13 所示。



图 7-13 添加了用户菜单项 Draw 的图形窗口

```
h1=uimenu(h1,'Label','&Gride on','callback','grid on');
```

此命令建立的菜单项为【Grid on】，快捷键为 Alt+g，回调程序【Grid on】用于给所画出的图形添加网格。

- Position

该属性用于调整菜单项的相对位置。

例如，在菜单【Draw】下增加一个新的子菜单项【Grid off】，比较和子菜单项【Grid on】的属性 Position 的值，在 MATLAB 命令窗口中键入：

```
h2_2=uimenu(h1,'Label','Gride o&ff','callback','grid off');
get(h2_1,'position')
```

ans =

1

get(h2_2,'position')

ans =

2

可以看出，属性Position的值等于菜单项的排列顺序，排在最上的子菜单【Gride on】的position值为1，排在第二位的【Gride off】子菜单的position值为2。当需要调整相对位置时，可以用set函数设置。

同样，图形窗口顶部菜单栏上横向排列的菜单也可以用Position属性来设置其相对位置，方法与纵向排列的菜单相同。例如，查询【Draw】菜单的位置，命令如下：

get(h1,'position')

ans =

6

- Checked和Separator

Checked属性用于设置是否在菜单项前添加选中标记，设为"on"表示添加，"off"表示不添加。

例如，在子菜单【Grid on】前添加选中标记，命令如下：

set(h2_1,'checked','on')

此时的图形窗口如图7-14所示。

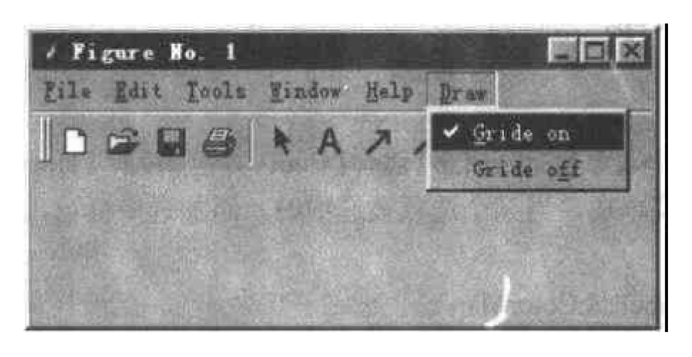


图7-14　用户菜单项Grid on添加选中标记的图形窗口

注意：有些菜单的选中标记相斥，【Gride on】和【Gride off】便属于这一类型。这两个菜单项不能同时为选中或不选中，这就要求给一个菜单项添加选中标记的同时去掉另一个菜单项的标记。

属性Separator用于在菜单项之前添加分隔符，以便使菜单更加清晰。

- BackgroundColor和ForegroundColor

这两个属性均为RGB向量，BackgroundColor（背景色）是菜单本身的颜色，ForegroundColor（前景色）是菜单上标注字符串的颜色。

下面通过一个实例来具体说明如何运用函数命令编写程序来制作包含菜单对象的用户界面。

【例7-4】建立一个只包含用户界面菜单项的图形界面，并可执行菜单项的相应功能，

分别绘制 Membrane、三维高斯分布 Peaks 和 sinc 函数（草帽图）图形。

为程序清晰起见，每段之前用注释语句说明。

```
%首先建立一个图形窗口，去掉窗口本身包含的菜单条和工具条，命名为 DrawGUI
h0=figure('MenuBar','none',...
    'toolbar','none',...
    'Name','DrawGUI');
```

例 7-4 中的省略号“...”（或者用三个以上的点“.”）在 MATLAB 语句中表示和下一行相连，这样可以把比较长的语句分开表示，或者把语句中每段功能比较独立的部分作为一行，以增加句子的可读性。

```
%从左至右依次建立各级菜单
%先建立【Draw】菜单下的【Membrane】、【Peaks】和【Sinc】子菜单项
h1=uimenu(h0,'Label','&Draw');
 h21=uimenu(h1,'Label','&Membrane',...
    'callback','membrane';'axis tight');
 h21=uimenu(h1,'Label','&Peaks',...
    'callback','peaks';'axis tight');
 h21=uimenu(h1,'Label','&Sinc',...
    'callback',...
['[x,y]=meshgrid(-5:0.5:5);',...
'r=sqrt(x.^2+y.^2)+ eps;',...
'z=sin(r)./r;',...
'mesh(x,y,z)']);
```

%其次建立第 2 个菜单【ColorMap】包含【Cool（冷色调）】、【Hot（暖色调）】和【Default（缺省值）】 3 个子菜单项。当某个选项被选中时，添加选中标记，同时去掉其他选项的选中标记

```
h1=uimenu(h0,'Label','&ColorMap');
h22(1)=uimenu(h1,'Label','&Hot',...
        'Accelerator','h',...
'Checked','on',...
'Callback',...
['set(h22,"Checked","off");',...
'set(h22(1),"Checked","on");',...
'colormap(hot)']);
  h22(2)=uimenu(h1,'Label','&Cool',...
        'Accelerator','c',...
'Callhack',...
['set(h22,"Checked","off");...
'set(b22(2),"Checked","on");'...
'colormap(cool)']);
```

```
h22(3)=uimenu(h1,'Label','&Default',...
        'Accelerator','d',...
'Callback',...
['set(h22,"Checked","off");...
'set(h22(3),"Checked","on");'...
'colormap("default")'];
```

注意：在单引号内的字符串必须用两个单引号（不等于双引号）表示所需的单引号。

%在设置选中标记时，先用命令 set(h22,"Checked","off")将向量 h22 中三个句柄对应的菜单项都设为未选中状态，然后，把选择的菜单项设为选中状态，以保证多个选项之间的互斥性

%最后建立控制坐标轴显示的菜单【Axis】，用于是否显示坐标轴，它和建立【ColorMap】菜单类似，添加选中标记，指定快捷键

```
h1=uimenu(h0,'Label','&Axis');
h23(1)=uimenu(h1,'Label','Axis o&ff',...
        'Accelerator','f',...
'Callback',...
['set(h23,"Checked","off");...
set(h23(1),"Checked","on");...
'axis off'];
   h22(2)=uimenu(h1,'Label','Axis o&n',...
        'Accelerator','n',...
['set(h23,"Checked","off");...
set(h23(2),"Checked","on");...
'axis on'];
```

至此，就形成了一个用户界面菜单，执行结果如图 7-15 所示。3 个菜单的子菜单项分别如图 7-16（a）、（b）、（c）所示。

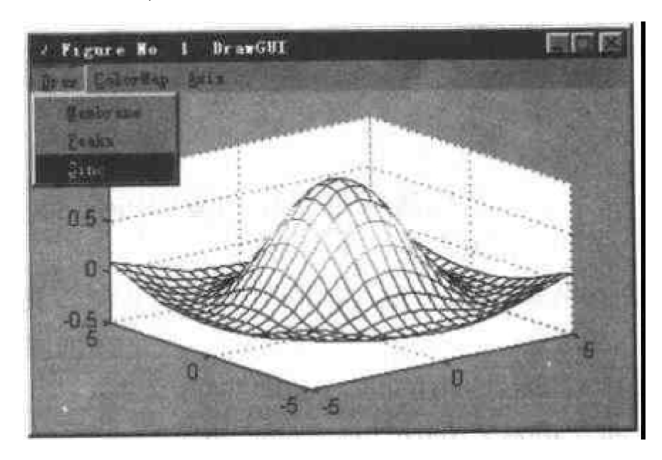
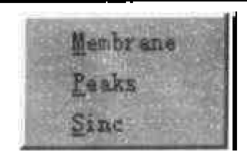
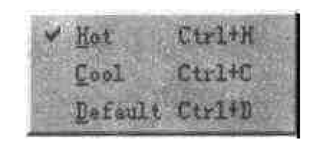


图 7-15　用户界面菜单程序执行后的结果



（a）【Draw】菜单项



（b）【Colormap】菜单项



（c）【Axis】菜单项

图 7-16　Draw、Colormap 和 Axis 菜单项

7.2.3　用户界面上下文菜单（Uicontextmenu）

从 MATLAB 5.3 版本开始，新增加了一个用户界面上下文菜单（Uicontextmenu）对象，与位置固定的菜单对象相比，上下文菜单对象的位置不固定，它总是与某个（些）图形对象相联系，并通过鼠标右键激活。制作菜单的步骤如下：

（1）利用函数 uicontextmenu 创建上下文菜单对象；

（2）利用函数 uimenu 为该上下文菜单对象制作具体的菜单项；

（3）利用函数 set 将该上下文菜单对象和某些图形对象联系在一起。

【例 7-5】绘制一条正弦曲线，并创建一个与之相联系的上下文菜单，用以控制正弦曲线的颜色。

建立 M 文件 C7L5 如下：

```
t=0:pi/50:2*pi;
y=sin(t);
h_line=plot(t,y);            %绘制正弦曲线，并返回句柄值 h_line
h=uicontextmenu;             %创建上下文菜单
uimenu(h,'label','red','callback','set(h_line,''color'',''r'')')    % 制作具体的菜单项，定义相
                                                                      应的回调
uimenu(h,'label','blue','callback','set(h_line,''color'',''b'')')
uimenu(h,'label','green','callback','set(h_line,''color'',''g'')')
set(h_line,'uicontextmenu',h)          %使 h 上下文菜单与正弦曲线 h_line 相联系
```

在 MATLAB 命令窗口中运行文件 C7L5.m，得到如图 7-17 所示蓝色正弦曲线。将鼠标指针指向线条，单击鼠标右键，弹出上下文菜单，在选中某菜单项后，正弦曲线将改变

为相应的颜色。

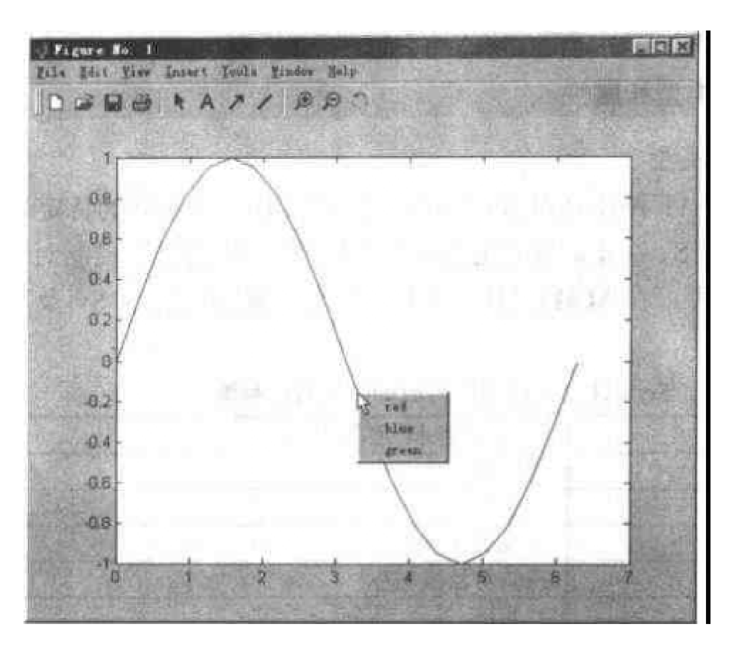


图 7-17　制作的用户上下文菜单

7.3　用户界面控制对象（Uicontrol）

除了菜单以外，控制对象是另一种实现用户与计算机交互的主要手段。用户界面控制对象（Uicontrol）是这样一类图形界面对象：用户用鼠标在控制对象上进行操作，单击鼠标时，将会使应用程序作出响应，并执行某些预定的功能子程序（Callback）。控制对象的操作结果是可见的，有时可以改变应用程序的初始状态。

7.3.1　控制对象的创建

和创建菜单对象类似，同样有两种方式用于创建控制对象：一是基于函数命令行的编程方式；二是基于 GUI 的方式。

1. *基于函数命令行的编程方式*

使用控制对象函数 uicontrol 是创建控制对象的基本方法，函数的调用格式为：

***h*=uicontrol('PropertyName1',value1,'PropertyName2,'value2,...)**

在当前图形窗口创建一个用户界面控制对象，并返回一个句柄值。

***h*=uicontrol(hfig,...)**

在特定的图形窗口创建一个用户界面控制对象。

其中，***h*** 为待制作的用户界面控制对象的句柄，hfig 为其父对象句柄，当 hfig 缺省时，MATLAB 将在当前图形界面上添加用户界面控制对象。

2. *基于 GUI 的方式*

直接在 MATLAB 命令窗口中输入 guid 命令，打开 GUI 设计工具集窗口，窗口的左

侧便是各种类型的控制对象按钮栏。单击要添加的控制对象，在图形窗口中“画出”所需要的位置和大小即可。

7.3.2 控制对象的类型和属性

1. 控制对象的类型

MATLAB 支持 11 种控制对象，每种控制对象对应于不同的特定目的。控制对象的类型由 Style 的属性值 Sv 决定。要创建某种控制对象，只需要将控制对象的 Style 属性值 Sv 设为所需要的值即可。在 MATLAB 中 11 种控制对象的属性值 Sv 及其对应的控制对象如表 7-3 所示。

表 7-3 Style 属性取值及所对应的控制对象类型

属性值 Sv	对应的控制对象的类型
Axis	坐标轴
Checkbox	复选框
Edit	可编辑文本框
Frame	控制对象区域框
Listbox	列表框
Popupmenu	弹出式菜单
Pushbutton	单功能按钮
Radiobutton	无线电选择按钮
Slider	滑动条
Text	静态文本框
Togglebutton	双位按钮

下面介绍 MATLAB 的 11 种控制对象的特征和实现的功能。

- 坐标轴（Axis） 设置坐标轴控制对象。
- 复选框（Checkbox） 也称为转换按钮，通常用于表明选项的状态或属性，由标志及标志左侧的小方框值成，当选中时，在小方框中填充“√”，Value 属性值设为 1；当没有选中时，方框为空，Value 属性值设为 0。
- 可编辑文本框（Edit） 一个凹形的方框，在屏幕上显示字符，并允许用户动态地编辑或重新安排文本，就像使用文本编辑器一样。
- 控制对象区域框（Frame） 仅是带色彩的矩形区域，在图面上用方框圈定用户控制对象所在的区域，一般用于组成按选按钮或其他用户控制对象。

注意：*在对象放入框架之前应事先定义框架，否则框架可能覆盖控制框，使它们不可见。*

- 列表框（List Boxes） 带有垂直滚动条的长方形文本框，列出字符串表，用户可以在这个列表中选取单个列表项和多个列表项。对应于选择，MATLAB 执行相应的功能和操作。
- 弹出式菜单（Popupmenu） 带有操作键的长方形文本框，一般用于向用户提出互斥的一系列选项清单，供用户选择。弹出式菜单不受菜单条的限制，可以位于图形窗口中的任何位置。
- 单功框值值（Pushbutton） 是其上带有文字标识的矩形。执行“按”、“放”

操作时会引起凹凸变化。一般用于执行一个动作而不是改变状态和属性。

- 无线电选择按钮（Radio Button）　带有文字标识的小圆圈，由标志及标志左侧的小圆圆组成。“开”、“关”状态用圆内的小黑点表示。当选中时，小圆圈被填充，Value 属性值设为 1；当没有选中时，圆圈为空，Value 属性值设为 0。无线电按钮一般用于在一组互斥的选项中选择一项。
- 滑动条（Slider）　亦称滚动条，包括三个独立的部分，分别是滚动槽或长方条区域，代表有效对象值范围；滚动槽内的指示器代表滑标当前值；以及滚动槽两端的箭头。一般用于从几个值域范围内选定一个。滑动条通常与所用文本控制对象一起显示标志、当前值及值域范围。滑动条的值用以下的方式设定：鼠标指针指向指示器，移动指示器。值动鼠标时要按住鼠标值，当指示器位于期望位置时，松开鼠标键；当指针处于槽中但在指示器的一侧时，单击鼠标键，指示器按该侧方向移动距离约等于整个值域范围的 10%；在滑动条不论哪一段上单击箭头，指示器沿着箭头的方向移动大约为滑动条范围的 1%。
- 静态文本框（Text）　仅仅是一个显示文本字符串的凹形方框，可使合法的 MATLAB 语句和程序输入运行，可随意输入运行指令程序。静态文本框一般用于显示标志、用户信息及当前值。

注意：*静态文本框之所以称为“静态”，是因为用户不能动态地修改所显示的文本，只能通过改变 String 的属性来更改。*

- 双位按钮（Togglebutton）　是其上带有文字标识的矩形。“开”和“关”状态用凹凸变化表示，两种状态切换选择使用。

2. 控制对象的几个重要属性

和其他图形对象一样，GUI 控制对象也有很多可以设置的属性（可参见 MATLAB 6.0 相应的帮助文件），本节将对其中一些重要的属性予以详细的介绍。

- Value 属性　控制对象的当前值，格式为标量或向量。该属性对不同的控制对象有不同的取值方式，分别为：
  - ◆ 复选框　被选中时，Value 的值为属性 Max 中设置的值；未选中时，Value 的值为属性 Min 中设置的值。
  - ◆ 列表框　被选中选项的序号，当有多个选项被选中时，Value 属性的他为向量。序号指的是选项的排列顺序，最上面的选项序号为 1，第二个选项序号为 2，…
  - ◆ 弹出式菜单　和列表框类似，也是被选中选项的序号，只是弹出式菜单只能有一个选项被选中，因而 Value 属性的值为标量。
  - ◆ 单选按钮　被选中时 Value 的值为属性 Max 中设置的值；未选中时 Value 的值为属性 Min 中设置的值。
  - ◆ 滑动条　Value 的值等于滑块指定的值。
  - ◆ 双位按钮　当双位按钮按下时，Value 的值为属性 Max 中设置的值；放开时，Value 的值为属性 Min 中设置的值。
  - ◆ 单功能按钮、可编辑文本框、区域框和静态文本框不设置这个属性的值。
- Max 属性　指定 Value 属性中可以设置的最大值，格式为标量。该属性对不同的

控制对象有不同的含义，分别为：

◆ 复选框　为复选框被选中 Value 属性的取值。

◆ 编辑框　如果 Max 的值减去 Min 的值大于，那么编辑框可以接受多行输入文本；如果小于等于 1，那么编辑框只能接受一行输入文本。

注意：只能输入一行文本时用 Enter 键结束输入；输入多行文本时用 Ctrl+Enter 键结束输入。

◆ 列表框　如果 Max 的值减去 Min 的值大于 1，允许选取多个选项；如果小于等于 1，只能选取一个选项。

◆ 无线电按钮　当无线电按钮被选中时的 Value 属性的取值。

◆ 滑动条　滑动条的最大值，缺省值是 1。

◆ 双位按钮　当按钮为"开"（被选中）时 Value 属性的取值。缺省值为 0。

◆ 文本框、弹出式菜单、单功能按钮和静态文本框不使用 Max 属性。

● Min 属性　指定 Value 属性中可以设置的最小值，格式为标量。该属性对不同的控制对象有不同的含义，分别为：

◆ 复选框　为复选框没有被选中的 Value 属性的取值。

◆ 编辑框　如果 Max 的值减去 Min 的值大于 1，编辑框可以接受多行输入文本；如果小于等于 1，编辑框只能接受一行输入文本。

◆ 列表框　如果 Max 的值减去 Min 的值大于 1，允许选取多个选项；如果小于等于 1，只能选取一个选项。

◆ 无线电按钮　为当无线电按钮没有被选中时的 Value 属性的取值。

◆ 滑动条　滑动条的最大值，缺省值是 1。

◆ 双位按钮　当按钮为"关"（未被选中）时 Value 属性的取值。缺省值为 0。

◆ 文本框、弹出式菜单、单功能按钮和静态文本框不使用 Min 属性。

下面将通过一个实例来具体说明如何运用函数命令编写程序来制作包含控制对象的用户界面。

【例 7-6】创建一个图形用户界面，使之包含静态文本、"无线电"选择开关、双选按键和控制对象区域框四种控制对象。

建立 M 文件 C7L6 如下：

```
clf reset
set(gcf,'menubar','none')                                    %在图形窗口中不设定菜单栏
set(gcf,'unit','normalized','position',[0.2 0.2 0.64 0.32]);  %给出当前图形对象的位置
set(gcf,'defaultuicontroluuits','normalized')             %设置用户控制对象缺省单位属性
h_axes=axes('position',[0.05 0.2 0.6 0.6]);              %绘制轴对象
t=0:pi/50:2*pi;y=sin(t);plot(t,y);
set(h_axes,'xlim',[0 2*pi]);
set(gcf,'defaultuicontrolhorizontal','left');
h_title=title('正弦曲线');
uicontrol('style','frame',...
'position',[0.67 0.55 0.25 0.25]);     %创建用户控制对象区域位置
```

```
uicontrol('style','text',...          %创建静态文本
'string','正斜体图名：',...
'position',[0.68 0.77 0.18 0.1],...
'horizontal','left');
hr1=uicontrol(gcf,'style','radio',...
'string','正体',...
'position',[0.7 0.69 0.15 0.08]);
%创建无线电选择按键，按键功能文字标识为“正体”，同时给定按键位置
set(hr1,'value',get(hr1,'Max'));      %因设置图名缺省值为正体，所以无线电按钮的小
                                      圆圈应被点黑
set(hr1,'callback',...
['set(hr1,"value",get(hr1,"Max"))',...
'set(hr2,"value",get(hr2,"min"))',...
'set(h_title,"fontangle","normal")']);     %选中将小圆圈 hr1 点黑，将互斥项 hr2 点白，
                                           并使图名字体正体显示
hr2=uicontrol(gcf,'style','radio','string','斜体',...
'position',[0.7 0.58 0.15 0.08]);
%创建无线电选择按键，按键功能文字标识为“斜体”，同时给定按键位置
set(hr2,'value',get(hr2,'Max'));
set(hr2,'callback',['set(hr2,"value",get(hr2,"Max"))',...
'set(hr1,"value",get(hr1,"min"))',...
'set(h_title,"fontangle","italic")']);     %使图名字体斜体显示
ht=uicontrol(gcf,'style','toggle',...
'string','Grid',...
'position',[0.67 0.40 0.15 0.12],...
'callback','grid');
%制作双位按钮“Grid”，并给出位置
```

运行结果如图 7-18 所示。当按下双位按钮【Grid】时，按钮呈凹状态，将在图内产生格栅，如图 7-19 所示。

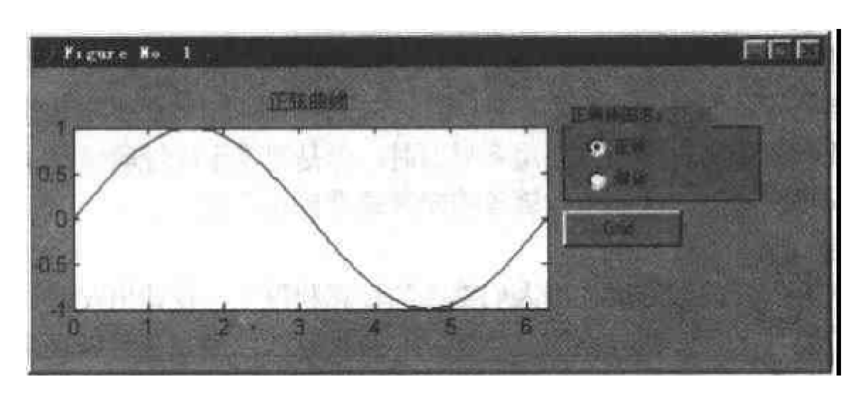


图 7-18　没加格栅的曲线图形

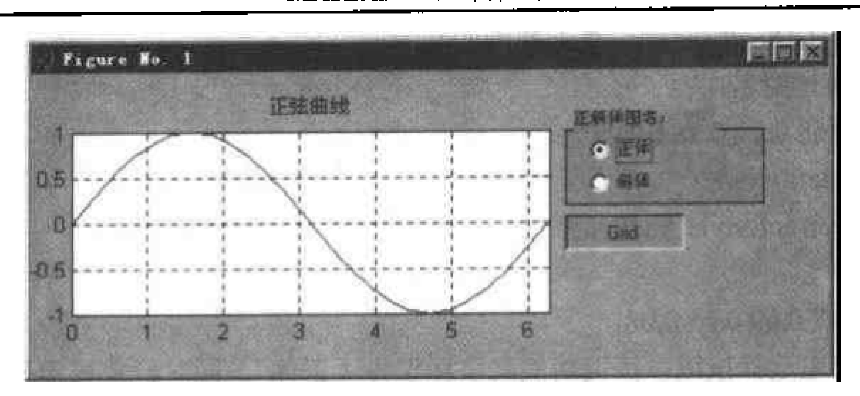


图 7-19　加上格栅的曲线图形

注意：区域框（Frame）仅为“醒目”起见而用，如果控制对象要用区域框围住，一定要先制作区域框，然后再生成被框围住的控制对象。否则，控制对象将被区域框遮住，因为区域框不透明；对于无线电按钮（Radiobutton），一组按键可用于表达那些互斥的选项；如某按键被选中，那么其他项将不能再选。

7.4　图形用户界面（GUI）设计

本节将介绍如何进行图形用户界面的制作，首先给出通常的 GUI 设计原则和设计步骤，然后具体讲解如何进行 GUI 设计。

7.4.1　图形用户界面的制作过程

1. GUI 设计原则

针对用户不同的需要，设计出的图形界面也各不相同。一般而言，一个较好的界面都遵循以下三个特性：简单性（Simplicity）、一致性（Consistency）和熟悉性（Familiarity）。

- 简单性

指在设计界面时，应力求简洁、直接、清晰地体现出界面的功能和特征。一些可有可无的功能应尽量删去，以保持界面的整洁。设计的图形界面要直观，因此应该多采用图形，而尽量避免数值。设计的界面应尽量减少窗口数目，力求避免在不同窗口之间进行来回切换。

- 一致性

一是用户自己开发的界面风格要尽量一致；二是新设计的界面要与其他已有的界面风格尽量保持一致。由于用户在初次使用新界面时，总是习惯于凭经验进行试探。例如，图形显示区通常安排在界面的左侧，按键等控制区通常放在右侧。

- 熟悉性

指在设计新界面时，应尽量使用人们熟悉的标志和符号，设计出友好、令人舒适的用户界面。用户可能并不了解新界面的具体含义和操作方法，但完全可以根据熟悉的标志作出正确地猜测，便于学习和使用。

除上述对界面的静态要求外，还应注意界面的动态性能。例如，所设计的界面对于用

户操作的响应要迅速（Immediate）和连续（Continous）；对持续时间较长的运算要给出等待时间提示，并允许用户中断等。

2. GUI 设计步骤

图形用户界面的制作一般包括界面设计和程序实现，对于初学者，建议按照下述步骤进行：

（1）分析界面所要求实现的主要功能，以明确设计任务。

（2）在纸上绘出界面草图，并站在使用者的角度审查草图。

（3）按照构思的草图，上机制作静态界面，并检查它。

（4）编写界面动态功能的程序，对功能进行逐项检查。

注意：以上所述的过程仅仅是一种建议，在设计中，步骤间也许要交叉或重复执行。由于设计和实现过程往往不是一步到位的，可能需要反复修改才能获得满意的界面。同时建议读者先进行界面布局编码，然后进行动态交互功能的编码。

7.4.2　GUI 设计工具集简介及其功能

从例 7-5 和例 7-6 可以看出，采用命令函数编写 M 文件来制作图形用户界面比较麻烦。MATLAB 出于减轻用户界面设计负担的考虑，自 MATLAB5.0 版本开始提供了一个功能强大的交互式制作用户界面的设计工具集，MATLAB 6.0 对其进行重新设计，给出了更为高效快捷的 GUI 设计工具集，即上文所述的基于 GUI 的创建对象的方式。GUI 设计工具集可以通过命令 guide 启动。其效果如图 7-20 所示。

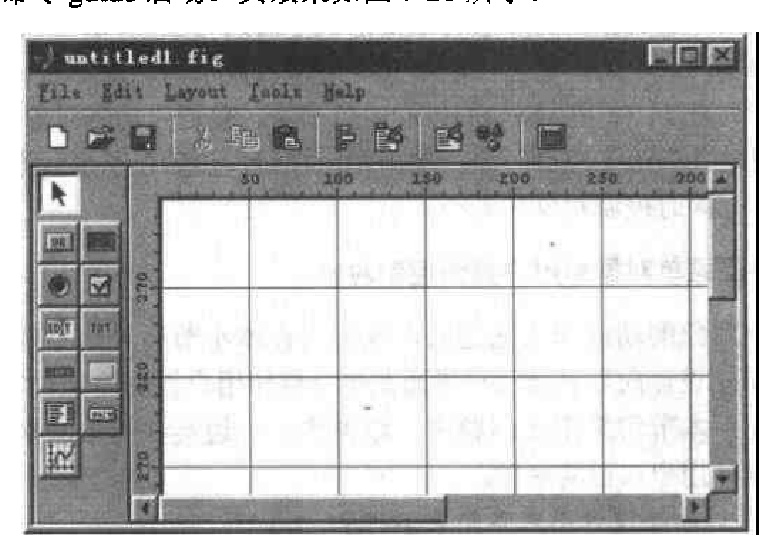


图 7-20　GUI 设计工具集

该窗口是一个 Windows 的标准应用程序窗口，同样具有菜单栏、工具栏以及工作区等栏目，包括了编辑图形属性、设计 GUI 菜单和 GUI 控制对象交互式等工具。

本节将着重介绍用于 GUI 设计的【Layout】菜单项和【Tools】菜单项。

- 【Layout】菜单项包含以下子菜单：
  - ◆ Align Object...　对齐对象；
  - ◆ Grid and Rulers　显示网格和标尺；

◆ Menu Editor　菜单编辑器，用于编辑和设计菜单。

- 【Tools】菜单项包含以下子菜单：
  ◆ Proerty Inspector 属性检查器；
  ◆ Object Browser 对象浏览器；
  ◆ Application Option 应用选项；
  ◆ Activate Figure 激活图形。

MATLAB 同时提供了用于启动“交互式编辑工具”的命令，如表 7-4 所示。

表 7-4　启动交互式编辑工具的命令表

命令	功能
guide	启动图形用户设计工具
propedit	启动图形对象属性编辑工具
menuedit	启动用户菜单属性编辑工具
align	启动图形对象几何位置排列工具
cbedit	启动用户菜单或控制对象的回调函数设计工具

注意：命令 menuedit 和 align 在 MATLAB 6.0 中已不再使用，如果用户使用这类命令，程序可照样运行，但系统将给出警告信息，同时给出 MATLAB 6.0 中相应的新命令函数。

在 MATLAB 工作窗口中运行表 7-4 中任何一种命令，都将会引出相应的交互式设计工具，进行相应的工作。但除了 guide 命令之外，其余命令所启动的编辑工具虽然同样可以在图形窗口中进行创建和修改等编辑工作，但是这种编辑结果仅仅存在于那个图形窗口中，一旦该窗口关闭，所有编辑的影响将全部消失。而只有在 guide 启动后，在受控状态下，图形窗口经编辑所产生的影响才可能以文件的形式被永久保留。

窗口的左侧陈列有 11 种可供添加的控制对象模块，包括坐标轴对象和（如“按键”、“弹出菜单”、“滑动条”等）控制对象。单击相应按钮，然后在图形区拖动鼠标，便可创建相应的控制对象，同时控制对象的大小。

7.4.3 设计用户界面菜单对象和用户界面控制对象

在介绍了控制对象的功能和用法之后，我们将在本小节通过两个例子来说明如何在图形窗口中利用 GUI 设计向导设计用户界面菜单对象和用户界面控制对象。

为了叙述方便，本节将采用在例题中一边说明，一边实践的方法来引导读者逐步掌握如何利用 GUI 设计用户自己的对象。

1. 利用 GUI 设计向导设计用户界面菜单对象

给图形窗口添加关于绘图的多层菜单项。

- 添加新菜单

首先，启动如图 7-20 所示设计工具集，同时打开一个新的图形窗口。然后单击工具栏上【Menu Editor】按钮、或者单击【Layout】‣【Menu Editor】命令启动菜单编辑器。单击在菜单编辑器最上面的【New Menu】按钮，此时在下方的列表区将产生一个代表新菜单的图标，右侧的编辑框呈可编辑状态，如图 7-21 所示。

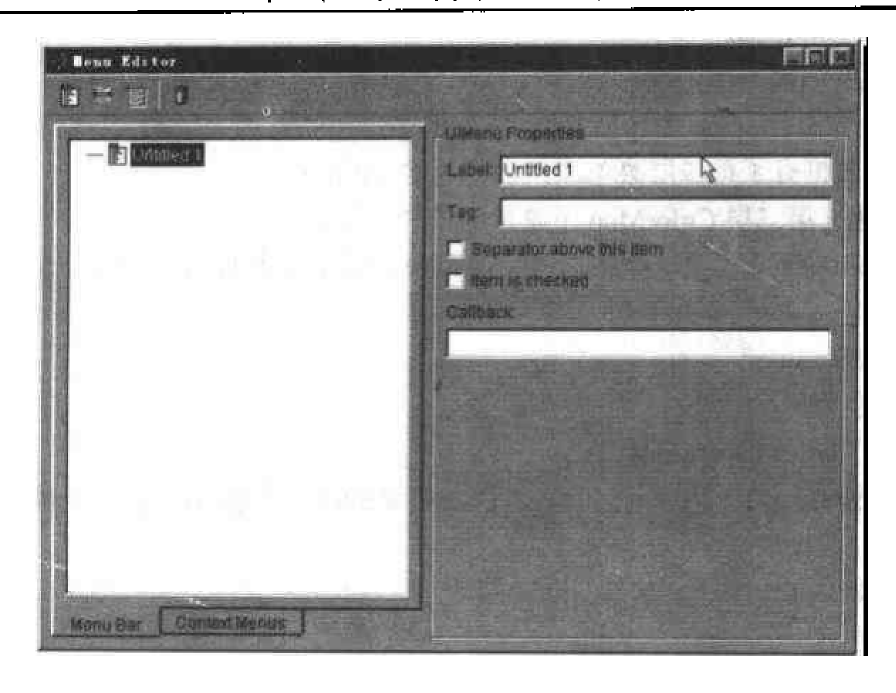


图 7-21 未编辑的新菜单项

完成一个菜单项的设置后，用相同的方法在适当的层次上可以继续建立菜单项。下面列出具体建立每一个菜单项的简要步骤和输入内容。

◆ 建立第一层菜单【Draw】

在图 7-21 所示对话框右侧的编辑框中依次输入下列各项：

Label：'&Draw'

Tag：'绘图'

Callback： ''

此时，左侧列表框中的菜单项名称变为 Draw。

◆ 建立第二层菜单

单击选中【Draw】菜单，然后单击【New Menu Item】按钮，将建立一个【Draw】的子菜单项，在各个编辑框中依次输入下列各项：

Label： '&Mesh'

Tag:：'三维网格'

Callback：' mesh(peaks); axis tight'

此时，列表框有了第二层菜单项，名称为 Mesh。

再次单击选中【Draw】菜单，然后单击【New Menu Item】按钮，在各个编辑框中依次输入下列各项：

Label： '&Contour'

Tag:：'三维等高线'

Callback：'contour3(peaks,20); axis tight'

此时，列表框有了第二层菜单项，名称为 Contour。

再次单击选中【Draw】菜单，然后单击【New Menu Item】按钮，在各个编辑框中依次输入下列各项：

Label：'&ColorMap'

Tag：'颜色映像'

Callback： ''

此时，列表框有了第二层菜单项，名称为 Colormap。

◆ 建立第二层 ColorMap 子菜单的第三层子菜单

单击【ColorMap】菜单，然后单击【New Menu Item】按钮。在各个编辑框中依次输入下列各项：

Label：'&Summer'

Tag：'暖色调'

Callback：'colormap(summer)'

单击【ColorMap】菜单，然后单击【New MenuItem】按钮。在各个编辑框中依次输入下列各项：

Label：'&Cool'

Tag：'冷色调'

Callback：'colormap(cool)'

单击【ColorMap】菜单，然后单击【New MenuItem】按钮。在各个编辑框中依次输入下列各项：

Label：'&Default'

Tag：'还原缺省色调'

Callback：'colormap("default")

至此，三层菜单均已设好。此时菜单编辑器的列表框中包括了所设计的菜单内容，如图 7-22 所示。

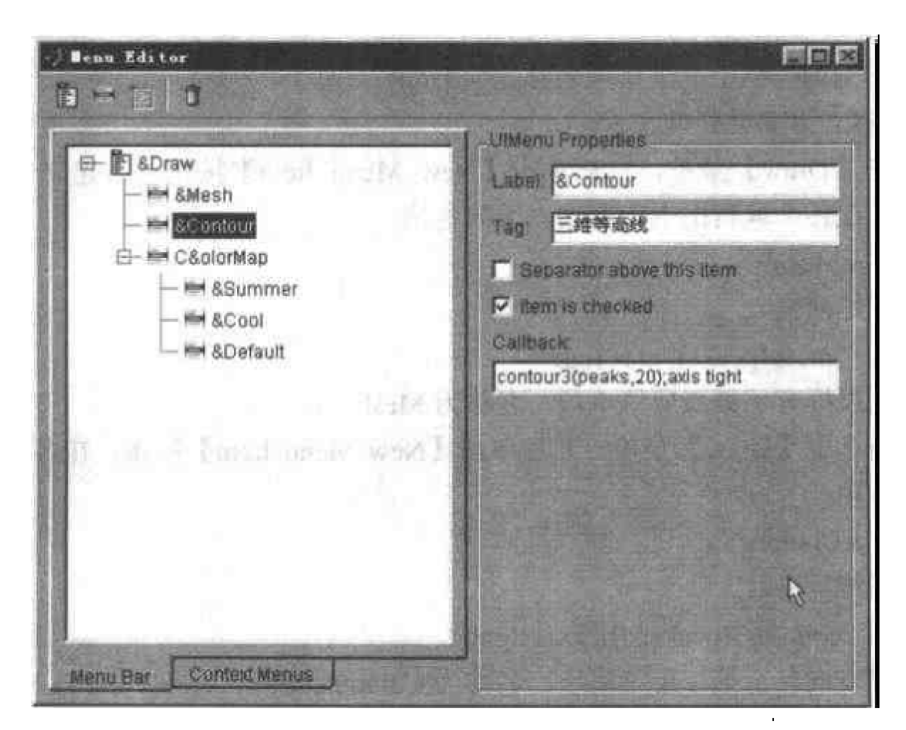


图 7-22　菜单设计完成后的列表框

关闭菜单编辑器，单击【Tools】▸【Activate Figure】命令，将图形窗口的状态改为活动状态，这时 MATLAB 弹出如图 7-23 所示对话框提示，在使图形窗口成为活动状态前保存图形。单击【Yes】按钮，输入“a1”将包括自己设计的菜单的图形窗口保存在名为“a1”的 M 文件中。

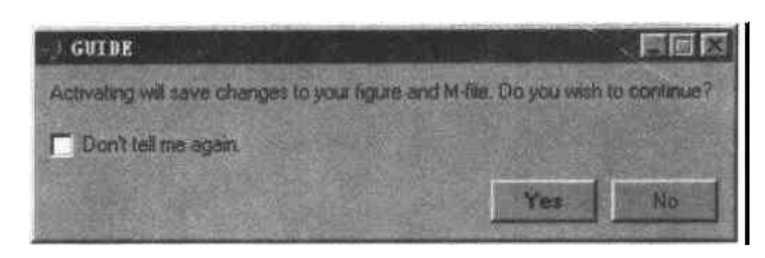


图 7-23　提示保存图形对象的对话框

至此，一个名为 a1 的包含有用户菜单的图形用户界面完全建成，并以 M 文件的格式保存在 MATLAB 的工作路径中。使用时，用户只要在 MATLAB 命令窗口中直接运行 a1，即可出现如图 7-24 所示的包含用户菜单项【Draw】的窗口。

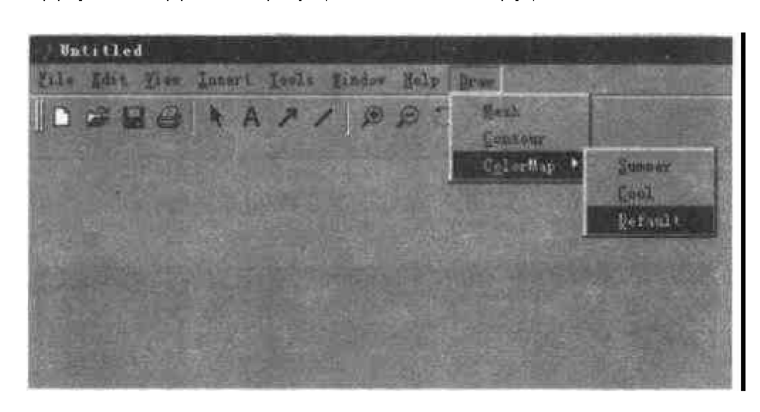


图 7-24　含有用户菜单项 Draw 的图形窗口

图形窗口成为活动状态后，新添加的菜单出现在该窗口中。单击【Draw】▸【Contour】命令，将会打开一个新的图形窗口，在该图形窗口中绘制三维等高线图形，如图 7-25 所示。

如果选择【Draw】▸【Mesh】命令，便会在图形窗口中绘制三维网格图，此时可用子菜单【Colormap】中的命令来调整图形颜色映象的色调，选择【Cool】选项，网格图的颜色将调整为冷色调，如图 7-26 所示。

● 对菜单进行进一步的处理

对于设计好的菜单经常需要进行进一步的处理。根据所调整内容的需要一般有两种不同的方法：

◆ 调整菜单项的顺序和层次关系、标注、标记以及回调程序等内容

对于此类的调整，可以和它们的建立一样，通过菜单编辑器完成。步骤如下：

（1）运行已经设计好的菜单的 M 文件，如前面建好的“a1”文件，显示出菜单所在的图形窗口。

（2）启动 GUI 设计工具集，将图形窗口设置为被控制（Controlled）状态，运行菜

单编辑器。

（3）从编辑器的列表框中选择需要调整的菜单项，要调整菜单项的顺序层次关系可以通过左侧的 4 个按钮完成；要调整菜单项的标注、标记以及回调程序等可以直接在下部的编辑框中进行。

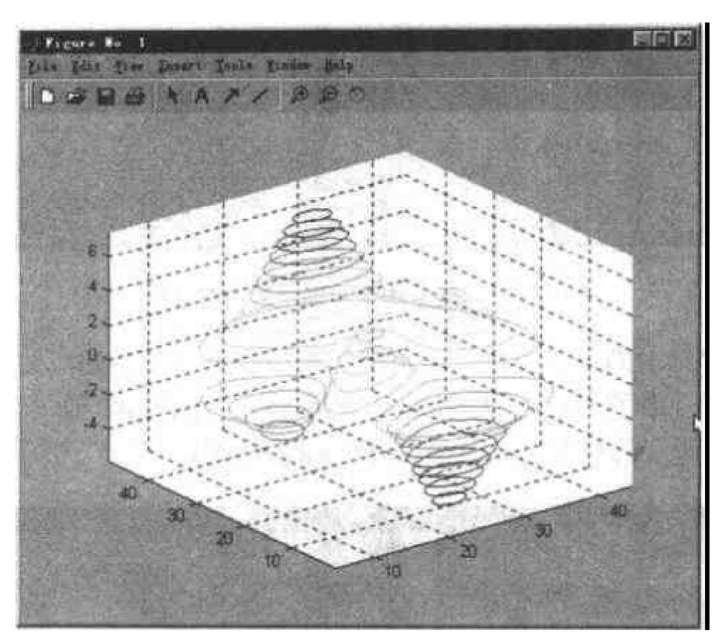


图 7-25　用户设计的菜单绘制等高线

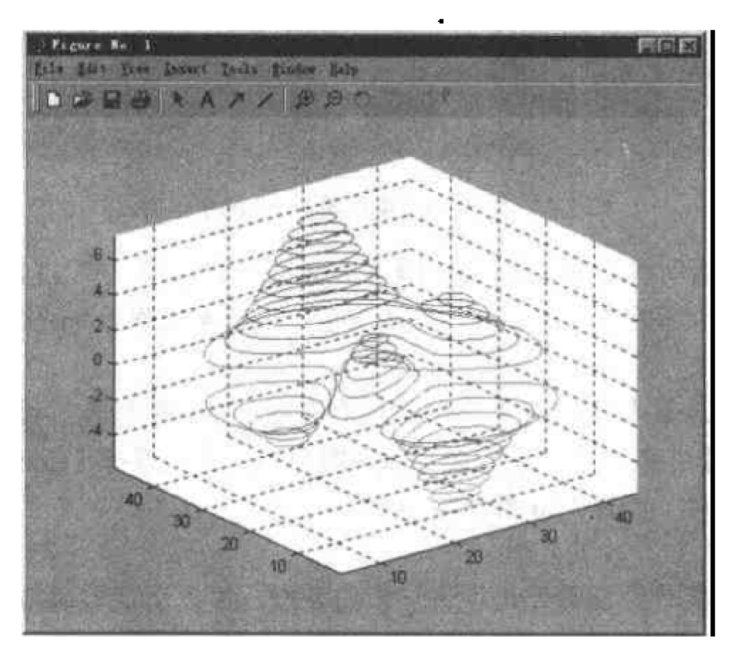


图 7-26　改变颜色映像

◆ 菜单项的其他属性

如果需要为菜单项添加加速键、分隔符，或在菜单项前标记为选中状态等，就需要借助于 7.3.1 中所讲的图形属性编辑器了。步骤如下：

（1）运行已经设计好的菜单的 M 文件，比如，前面建好的“a1”文件，显示出菜单

所在的图形窗口。

（2）启动图形属性编辑器，选中要编辑的菜单项，修改相应的属性。

菜单的 Separator 属性用于设置是否在该菜单项前添加分隔符，“on”为添加分隔符，off 为不添加。Checked 属性用于设置是否在该菜单项前添加选中标记，“on”为添加，off 为不添加。

下面对前面的菜单项进行进一步修改，使得在选中的菜单项前添加选中标记，并将有子菜单的【ColorMap】项与其他项用分隔符分隔开来。

在 MATLAB 的命令窗口中直接运行“a1”文件，显示出用户菜单所在的图形窗口，启动菜单编辑器，选中要编辑的菜单项，按照本节所述修改相应的属性。

修改后的用户菜单项如图 7-27 所示，可以看出【Mesh】和【Contour】项前都加了选中标记，并且【ColorMap】菜单项与【Mesh】和【Contour】菜单项之间有分隔符（一条凹线）。

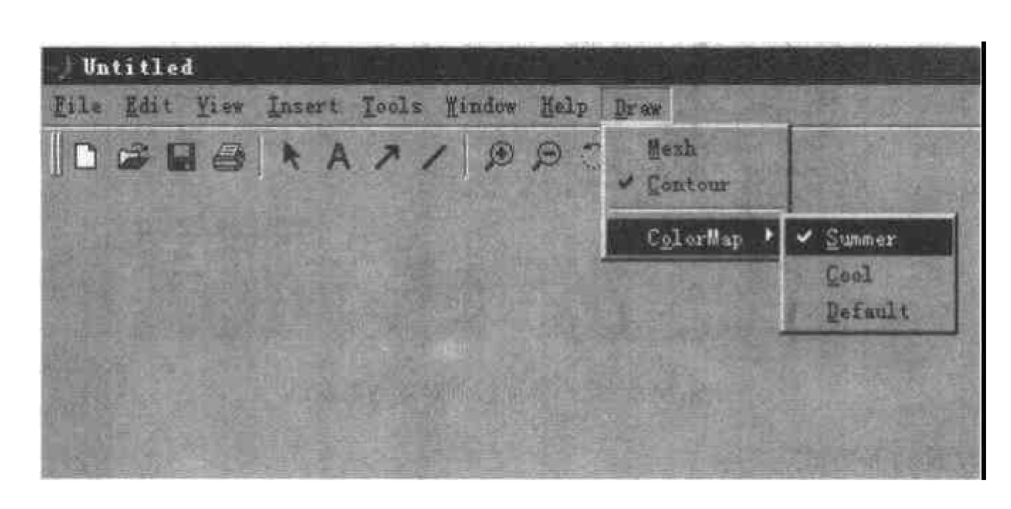


图 7-27　经过处理后的菜单项

2. 利用 GUI 设计向导设计用户界面控制对象

利用 GUI 设计工具集可以实现多种控制对象的设计，本部分将介绍如何在图形窗口中建立控制对象，以得到良好的图形用户界面，增强界面的交互能力。

在 GUI 设计工具集中，单击需要添加的控制对象图标，双击图标，即可弹出图形属性检查器，显示出该控制对象的属性，同时可以对该对象进行编辑。

在前面所讲述的 peak 图形窗口中添加控制对象。

● 建立控制对象并设置基本属性

在 MATLAB 命令窗口中运行 a1 命令，调出图 7-24 所示包含用户菜单项的图形窗口，启动设计工具集。

现在将要设计的控制对象图标——三个单选按钮、一个静态文本框和一个弹出式菜单对象复制到图形窗口下方的适当位置，并调整至合适的尺寸。然后通过双击图形窗口中控制对象的图标来启动属性检查器，从面编辑它们的属性。

三个按钮分别用于绘制网格图、等高线和退出操作。在 String 属性中应该分别输入“绘制网格图”、“绘制等高线”和“退出操作”，这些字符串将作为该按钮的标注，同时在属

性 FontSize 中设置字体大小为“12”（以下控制对象的字体大小属性统一为 12，不再说明）。

弹出式菜单列有几种颜色映象的色调供用户选择，因而在属性 String 中输入“缺省色调|冷色调|暖色调”字符，作为弹出式菜单的选项。

静态文本框置于弹出式菜单的上方作为提示所用，在 String 属性中输入文本框的文本“选择色调:”。缺省状态下，控制对象的标注文本都是居中排列，如果希望文本靠左排列，便将属性 HorizontalAlignment 设为“left”。

将这些属性设好后保存，并激活图形对象，此时的图形用户界面如图 7-28 所示。此时还不能利用这些按钮来对图形进行具体操作，如果想要执行这些控制对象所代表的含义，必须编写相应的回调程序，这将在第 3 部分进行说明。

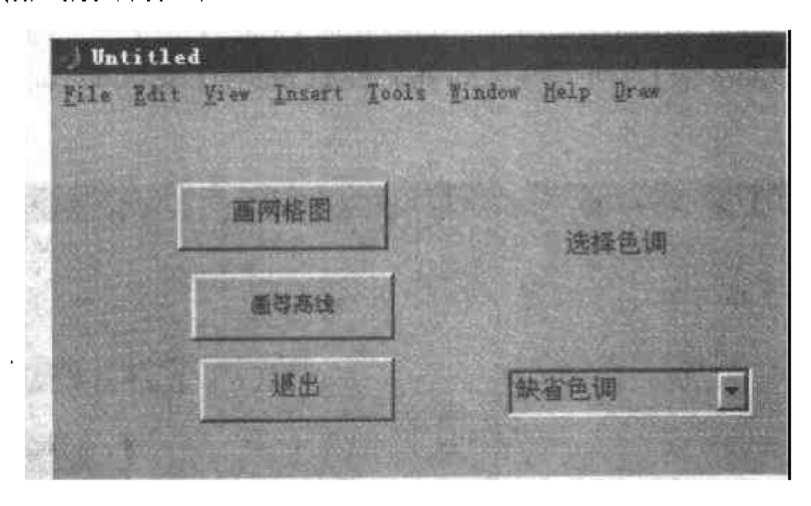


图 7-28　用户界面控制对象图形窗口

● 设置对齐方式

由于手工排列的控制对象不易真正对齐，为此，MATLAB 在设计工具集中专门提供了一个用于排列对象的工具按钮【Alignment Object】。单击该图标，或者单击【Tools】▸【Alignment Object】命令，将会出现如图 7-29 所示的“Align Objects”对话框。

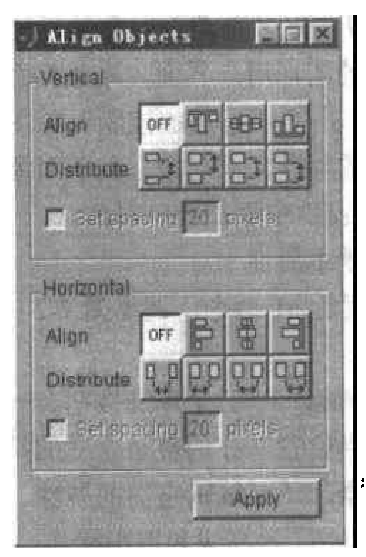


图 7-29　排列工具对话框

对话框上方的窗口显示在图形窗口中可以进行排列操作的对象，在此选择要处理的对象，连续拖动鼠标可进行对象的多选。

对话框中有两种类型的图标：Align、Distribute，还有 Set Spacing 编辑框，均分为竖直和水平两种情况。

- ◆ Align　水平和竖直方向共6个按钮，用于设置对象的对齐方式；
- ◆ Distribute　水平和竖直方向共8个按钮，用于使对象等间距分布；
- ◆ Set Spacing　两个编辑框用于指定对象的水平或竖直间距，以"点"为单位指定数值。

对话框下部的【Apply】按钮将设置的排列方式应用于选中的对象。

如果想令三个按钮之间排列整齐，在选中三个按钮对象之后，首先单击"Vertical"框中 Distribute 下的第一个图标，使三个按钮之间上下间隔相等，并单击【Apply】使格式生效；然后单击 Align 下的第二排第一个图标，使按钮靠右对齐。

采用同样的方法将静态文本框和弹出式菜单排列整齐，对齐后的图形用户界面如图7-30所示。

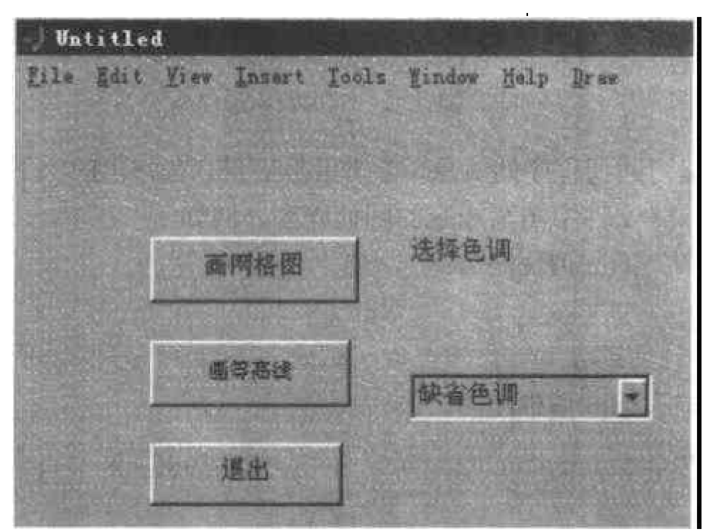


图7-30　对齐后的图形用户界面

● 编写回调程序

控制对象和菜单对象一样可以在属性 Callback 中输入回调程序，本部分将介绍如何利用 GUI 设计工具集中的"Callback Editor"（回调程序编辑器）工具来编写回调程序。

选中要编写回调程序的控制对象，单击鼠标右键，在弹出的上下文菜单中选择【Edit Callback】选项，系统将自动调用该图形用户界面的 M 文件，并将光标自动定位在回调子程序处，如图7-31所示。

对于不同的事件（Event），有不同类型的回调程序，而且每一类图形对象包括的事件种类也不同。并非每一种事件都有自己的"Callback"程序。

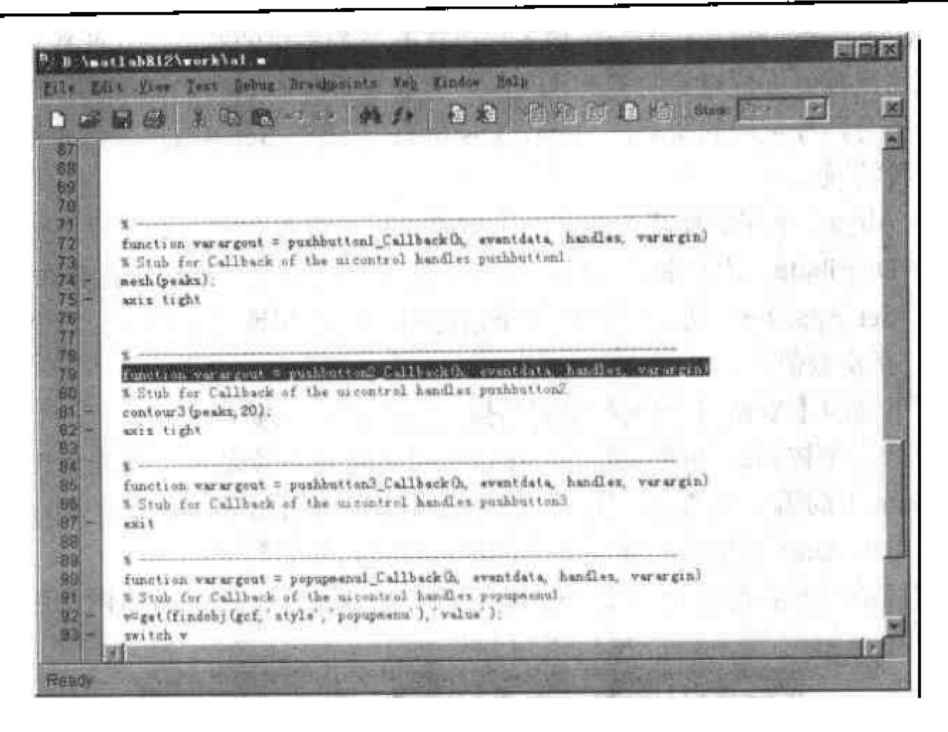


图 7-31　回调程序编辑器

首先，选中要编写回调程序的对象，单击鼠标右键，在弹出的上下文菜单中选择【Edit Callback】选项，在 M 文件中填写相应的回调程序。例如

按钮“画网格图”的回调程序为：

```
mesh(peaks);
axis tight
```

按钮“画等高线”的回调程序为：

```
contour3(peaks,20);
axis tight
```

弹出式菜单的回调程序为：

```
v=get(findobj(gcf,'style','popupmenu'),'value');
switch v
    case 1
    colormap('default')
    case 2
    colormap(cool)
    case 3
    colormap(summer)
end
```

在弹出式菜单的回调程序中，用 findobj 函数来获取弹出式菜单的句柄。函数 get 用于获取控制对象的 Value 属性的值，即用户选择的选项序号，命令 switch 用该序号来控制各个选项和回调程序的对应关系。

在图形对象的属性检查器内的 Name 属性项中，输入图形用户界面的名称，如 My

GUI，至此，便完整地建立了一个图形用户界面。该界面包括用户界面菜单项【Draw】和用户界面控制对象。用户此时即可通过任意一种方式，或者菜单项，或者控制按钮，来控制程序的执行，绘制相应的曲线，并选择特定的颜色映像色调。例如，要画出暖色调的网格图，除了利用菜单项外，可以在弹出式菜单下选取【暖色调】选项，单击【画网格图】按钮即可，结果如图 7-32 所示。

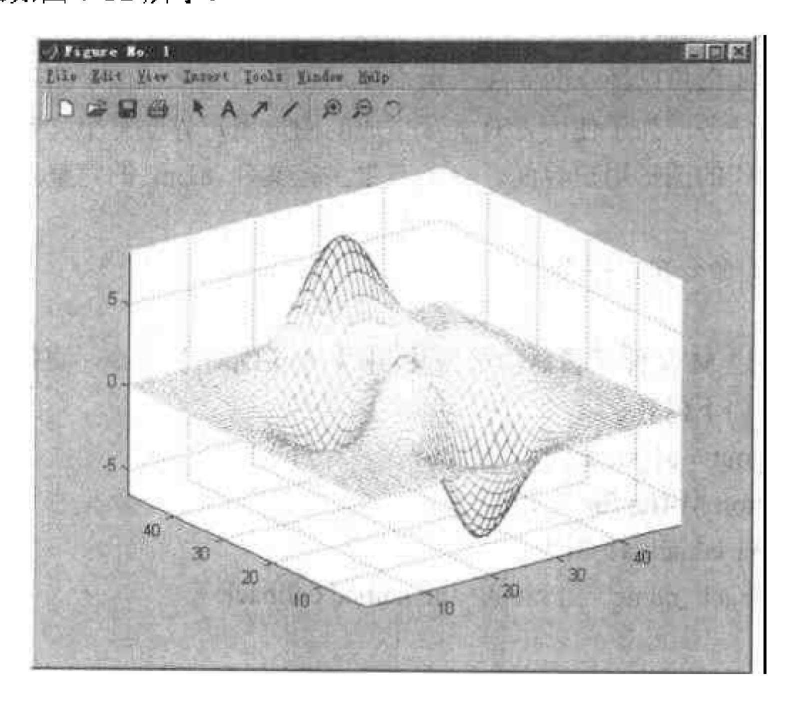


图 7-32　设计完整的图形用户界面

单击【退出】按钮结束对图形窗口的操作，关闭界面。

在本节中利用 GUI 设计向导设计了一个用户自己的图形用户界面，其中包括菜单和控制对象。可以看出，利用 GUI 设计向导制作 GUI 非常方便快捷。MATLAB 将图形用户界面以 M 文件的形式保存在当前目录下。实际上 GUI 设计向导的作用是根据用户的设计产生相应功能的 M 文件，然后运行这个 M 文件生成相应的用户界面。

7.4.4 用户图形界面功能的测试和配套文件

1. 对界面功能的测试

由于回调程序只有在被调用时才可以被 M 文件解释，因此要判断回调程序是否正确，必须对每项功能进行逐项调试。在测试中，要尽量发现运行结果和要求不相符的部分，并对相应的软件加以修改。切记一定不要以界面的外形武断地下结论，也不要以个别功能的观察结果代替全面的测试。

就本章制作的图形用户界面而言，应该进行以下几项测试：

- 界面的缩放测试。观察各个对象能否保持相对几何关系基本不变。

编辑框接受指令能力的测试。可分为如下类型：

◆ 单行指令、多行指令；
◆ 二维平面图形指令；
◆ 静态图形指令。

- 其他控制对象能力的测试，包括如下几项：
  - 缺省设置是否合理；
  - 变化控制对象选项时，界面表现是否正常；
  - 是否有多项选择的能力；
  - 菜单能力的测试。

2. 为用户提供的配套文件

一般而言，至此用户已经全面完成了用户图形界面的制作。由于图形用户界面设计本身是个比较复杂的任务，为了使读者对本章生成的图形用户界面有个完整的认识，下面给出由机器根据所制作的图形用户界面，自动产生主控文件 a1.m 的完整内容和相应的配套数据。

在 MATLAB 的命令窗口中键入：

```
type a1
```

用户也可以通过 M 文件编辑器，单击【File】▸【Open】命令，直接打开 a1.m 文件。屏幕上将显示如下内容：

```
function varargout = a1(varargin)
% A1 Application M-file for a1.fig
%     FIG = A1 launch a1 GUI
%     A1('callback_name', ...) invoke the named callback

% Last Modified by GUIDE v2.0 14-Mar-2001 18:15:27

if nargin == 0   % LAUNCH GUI

fig = openfig(mfilename,'reuse');

% Use system color scheme for figure
set(fig,'Color',get(0,'defaultUicontrolBackgroundColor'));

% Generate a structure of handles to pass to callbacks, and store it
handles = guihandles(fig);
guidata(fig, handles);

if nargout > 0
     varargout{1} = fig;
end

elseif ischar(varargin{1}) % INVOKE NAMED SUBFUNCTION OR CALLBACK

try
```

```
    [varargout{1:nargout}] = feval(varargin{:}); % FEVAL switchyard
catch
    disp(lasterr);
end

end


% ABOUT CALLBACKS
% GUIDE automatically appends subfunction prototypes to this file, and
% sets objects' callback properties to call them through the FEVAL
% switchyard above. This comment describes that mechanism
%
% Each callback subfunction declaration has the following form
% <SUBFUNCTION_NAME>(H, EVENTDATA, HANDLES, VARARGIN)
%
% The subfunction name is composed using the object's Tag and the
% callback type separated by '_', e.g. 'slider2_Callback'
% 'figure1_CloseRequestFcn', 'axis1_ButtondownFcn'
%
% H is the callback object's handle (obtained using GCBO)
%
% EVENTDATA is empty, but reserved for future use
%
% HANDLES is a structure containing handles of components in GUI using
% tags as fieldnames, e.g. handles.figure1, handles.slider2. This
% structure is created at GUI startup using GUIHANDLES and stored in
% the figure's application data using GUIDATA. A copy of the structure
% is passed to each callback.  You can store additional information in
% this structure at GUI startup, and you can change the structure
% during callbacks.  Call guidata(h, handles) after changing your
% copy to replace the stored original so that subsequent callbacks see
% the updates. Type "help guihandles" and "help guidata" for more
% information.
%
% VARARGIN contains any extra arguments you have passed to the
% callback. Specify the extra arguments by editing the callback
% property in the inspector. By default, GUIDE sets the property to:
% <MFILENAME>('<SUBFUNCTION_NAME>', gcbo, [], guidata(gcbo))
```

```
% Add any extra arguments after the last argument, before the final
% closing parenthesis.




% --------------------------------------------------------------------
function varargout = pushbutton1_Callback(h, eventdata, handles, varargin)
% Stub for Callback of the uicontrol handles.pushbutton1
mesh(peaks);
axis tight

% --------------------------------------------------------------------
function varargout = pushbutton2_Callback(h, eventdata, handles, varargin)
% Stub for Callback of the uicontrol handles.pushbutton2
contour3(peaks,20);
axis tight

% --------------------------------------------------------------------
function varargout = pushbutton3_Callback(h, eventdata, handles, varargin)
% Stuh for Callback of the uicontrol handles.pushbutton3
exit

% --------------------------------------------------------------------
functiou varargout = popupmenu1_Callback(h, eventdata, handles, varargin)
% Stuh for Callback of the uicontrol handles.popupmenu1
v=get(findobj(gcf,'style','popupmenu'),'value');
switch v
    case 1
    colormap('default')
    case 2
    colormap(cool)
    case 3
    colormap(summer)
end

>>
```

通过显示的内容可以看出，机器自动生成的主控文件的整体风格和一般手工编制的是不同的。文件起始部分的注释是关于这种机器产生文件的一般性说明，一般有固定的格式，

变换不大。文件中创建对象时的句柄变量名有如下特征：

- 通常是 *h0*、*h1*、*h2* 等名称；
- 同一变量名常被重复赋值，因此，它们的有效性是短暂的；
- *h0*、*h1*、*h2* 通常被用于表示“图形窗”、“子对象”、“孙对象”的句柄；
- 如果不加特别限制，每个对象都自动拥有一个“假名”。如有必要，用户也可以人工输入，但一定要保证惟一性。

由于机器自动生成了 a1.m 的主控 M 文件，因此用户只要在命令窗口中直接键入 a1，便可以生成用户所建立的图形用户界面。

7.5　用户界面对话框设计

目前，几乎所有的 Windows 程序都需要借助于对话框与用户进行人机对话和信息交流。所谓对话框是指一种带有各种控制对象的用户界面，一般是弹出显示的单独窗口，用来要求或提供信息。本节将就如何设计和制作用户界面对话框作一定的介绍。

7.5.1　专用对话框设计

从理论上讲，每个对话框都可通过编程从最基本的图形窗口开始逐步建立，但这样的工作量很大。由于许多用于专门用途的对话框格式都很固定，因而 MATLAB 提供多种专门建立专用对话框的函数。这样，就使得界面比较规范和统一，也给用户带来很大方便。MATLAB 用于建立专门对话框的函数，如表 7-5 所示。

表 7-5　MATLAB 建立专门对话框的函数列表

函数	所建对话框类型
errordlg	错误信息提示对话框
helpdlg	帮助文件对话框
inputdlg	输入信息对话框
pagedlg	设置图形位置对话框
printdlg	打印对话框
questdlg	提问对话框
warndlg	警告信息对话框
msgbox	消息框

下面，分别介绍这些函数的用法。

- errordlg 函数

其调用格式如下：

- ◆ errordlg　建立并显示一个错误信息对话框；
- ◆ errordlg（'errorstring'）　建立并显示一个包含字符串“errorstring”的错误信息对话框；
- ◆ errordlg（'errorstring','dlgname'）　建立并显示一个名为“dlgname”包含了字符串“errorstring”的错误信息对话框；

◆ errordlg('errorstring','dlgname','on')指定是否代替已经存在的同名对话框，"on"表示将已有的对话框弹出显示，不建立新的对话框；"off"则是建立一个新的对话框。

例如，下面的命令将产生如图 7-33 所示的错误对话框。

errordlg('Invalid Operation!!','error dialog')



图 7-33 错误对话框

● helpdlg 函数

其调用格式如下：

◆ helpdlg 显示一个缺省帮助对话框；

helpdlg('helpstring') 显示一个包含字符串"helpstring"的帮助对话框；

◆ helpdlg('helpstring','dlgname') 显示一个名字为'dlgname' 包含字符串'helpstring'的帮助对话框；

◆ h = helpdlg(...) 返回对话框的句柄值。

例如，下面命令的执行结果如图 7-34 所示。

helpdlg('Choose 10 points from the figure','Point Selection');

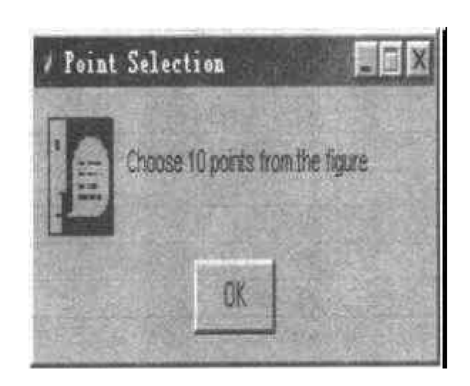


图 7-34 帮助对话框

● inputdlg 函数

其调用格式如下：

◆ answer = inputdlg(prompt) 建立一个输入对话框，用户输入返回到"answer"，"prompt"是提示字符串；

◆ answer = inputdlg(prompt,title) 建立一个标题为"title"的输入对话框；

◆ answer = inputdlg(prompt,title,lineNo) 建立一个标题为"title"的输入对话框。

"lineNo" 用于指定用户输入值的行数；

- ◆ answer = inputdlg(prompt,title,lineNo,defAns)　"defAns" 用于指定每个输入项的缺省值，格式为字符串；
- ◆ answer = inputdlg(prompt,title,lineNo,defAns,Resize)　Resize 用于指示对话框能否被改变大小，包括 "on" 和 "off" 两种取值，分别表示可以改变和不可改变。缺省情况下输入对话框大小不能改变，它是有模式的对话框；如选中"on"，则对话框自动成为无模式对话框。

所谓有模式对话框，指在对话框被关闭前，用户无法在程序的其他地方进行工作，例如 "Open File" 对话框就是典型的有模式对话框；无模式对话框在保留在屏幕上的同时，用户还可以在程序的其他窗口进行工作，"Find" 对话框是典型的无模式对话框。

例如，下面命令的执行结果将建立如图 7-35 所示输入对话框。

```
prompt= {'Enter matrix size:','Enter colormap name:'};
title= 'Input for peaks function';
lines= 1;
def = {'20','hsv'};
answer = inputdlg(prompt,title,lines,def);
```

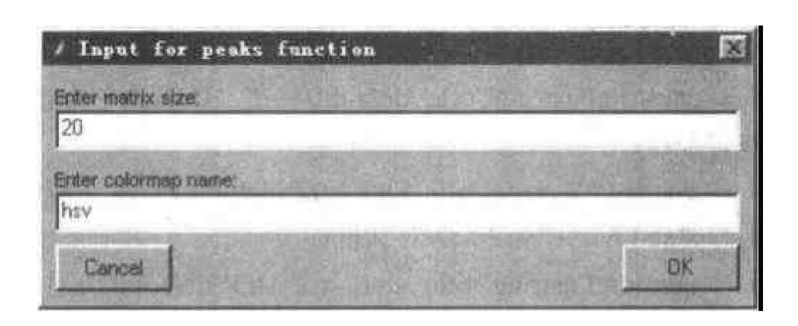


图 7-35　输入对话框

- pagedlg 函数

其调用格式如下：

- ◆ pagedlg　显示当前图形的页面位置对话框，用户可以通过该对话框来对当前的页面属性进行设置；
- ◆ pagedlg(fig)　显示指定句柄的图形窗口页面位置对话框。

若当前窗口中没有图形窗口，那么该函数会自动建立一个缺省的图形窗口。

例如，命令 pagedlg(gcf)将建立一个图形窗口的页面位置对话框，如图 7-36 所示。

- printdlg 函数

其调用格式如下：

- ◆ printdlg　显示当前窗口的打印对话框；
- ◆ printdlg(fig)　建立指定句柄图形窗口的打印对话框；
- ◆ printdlg('-crossplatform',fig)　显示标准的跨平台打印对话框而不是 Windows 平台内置的打印对话框，该对话框不受操作系统平台的限制。

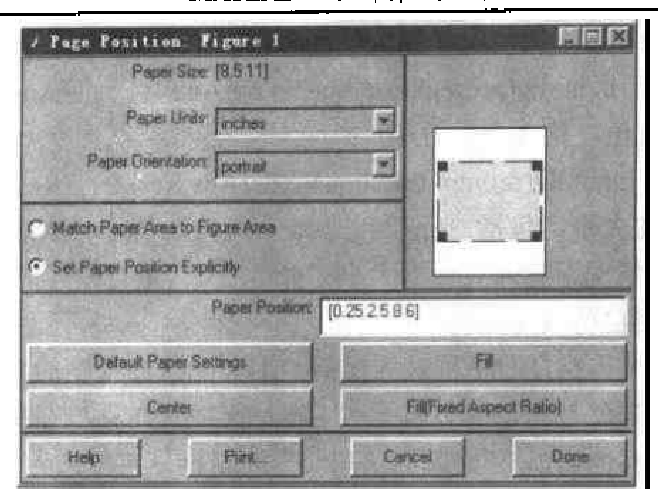


图 7-36 页面位置调整对话框

- questdlg 函数

其调用格式如下：

- ◆ button = questdlg('qstring') 建立提示问题为“qstring”的提问对话框。对话框有三个缺省的按钮【Yes】、【No】、【Cancel】。button 包含所按下的按钮的名字；
- ◆ button = questdlg('qstring','title') 为对话框加上标题“title”；
- ◆ button = questdlg('qstring','title','default') 在“Default”中指定当按下 Enter 键时的缺省按钮；
- ◆ button = questdlg('qstring','title','str1','str2','default') 建立有两个指定按钮、标注分别为“str1”、“str2”的对话框；
- ◆ button = questdlg('qstring','title','str1','str2','str3','default') 建立有三个按钮的对话框，按钮标注分别为“str1”、“str2”以及“str3”，缺省按钮为【Default】。

- warndlg 函数

其调用格式如下：

- ◆ warndlg 建立一个标题为“Warning Dialog”包含字符串 'This is the default warning string.' 的警告对话框；
- ◆ warndlg('warningstring') 用 warningstring. 指定警告字符串；
- ◆ warndlg('warningstring','dlgname') 用 warningstring. 指定警告字符串，用 dlgname 指定对话框标题；
- ◆ h = warndlg(...) 返回对话框句柄。

例如，下述命令将建立如图 7-37 所示警告对话框。

warndlg('Pressing OK will clear memory','!! Warning !!')

- msgbox 函数

其调用格式如下：

- ◆ msgbox(message) 建立一个消息框，显示 message 指定的信息。message 可以是字符串的向量、矩阵或数组；
- ◆ msgbox(message,title) 用 title 指定消息框的标题；

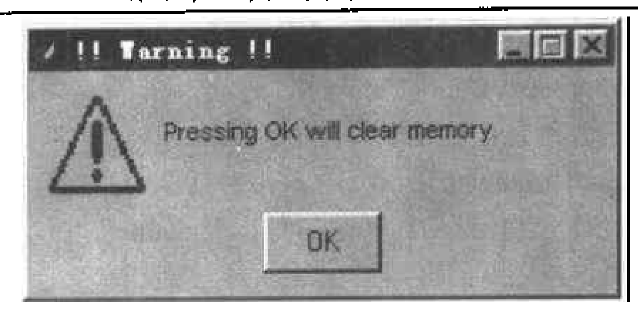


图 7-37　警告对话框

- ◆ msgbox(message,title,'icon')　用 icon 指定在消息框中使用的图标。icon 的取值有 none、error、help、warn 或 custom，分别对应不显示图标、错误图标（错误对话框中的图标）、帮助图标（帮助对话框中的图标）、警告图标（警告对话框中的图标）或自定义图标，缺省情况为"none"；
- ◆ msgbox(message,title,'custom',iconData,iconCmap)　自定义一个图标，"iconData"为定义图标的数据，"iconCmap"为图标采用的颜色映像；
- ◆ msgbox(...,'createMode')　指定消息框是有模式还是无模式对话框。createMode 的取值包括 modal（有模式）和 non-modal（无模式）。不指定该参数时消息框是无模式对话框；
- ◆ h = msgbox(...)　返回消息框的句柄。

7.5.2　标准对话框

在通常情况下，标准对话框都是 Windows 下内置的资源，只要使用相应的函数就能调用。例如，一般的应用软件都有【File】菜单项，及其下所含的【Open】、【Save】、【Save as】和【Exit】等选项，都是标准对话框。MATLAB 为用户提供了调用标准对话框的函数，通过这些函数，用户可以方便地为自己建立的菜单选项加上相应的标准对话框作为回调程序。

MATLAB 的调用标准对话框的函数如表 7-6 所示。

表 7-6　调用标准对话框的函数

函数	所建的对话框类型
uigetfile	打开已有的文件对话框
uipitfile	保存文件对话框
uisetfont	设置字体对话框
uisetcolor	设置颜色对话框

下面详细介绍这些函数的调用格式及含义。

- ● uigetfile 函数

函数的调用格式包括：

- ◆ uigetfile　显示一个"打开文件"对话框，该对话框自动列出当前路径下的目录和具有缺省扩展名".m"的文件；
- ◆ uigetfile('FilterSpec')　"FilterSpec"指定初始时显示的文件名或文件类型，可以使用通配符"*"；

◆ uigetfile('FilterSpec','DialogTitle') DialogTitle 指定对话框的标题；

◆ uigetfile('FilterSpec','DialogTitle',x,y) 指定对话框在屏幕上的位置，x、y 分别是到屏幕左边和上边的距离，单位是像素。

◆ [fname,pname] = uigetfile(...) 在单击对话框中的执行按钮后，返回选择的文件名和路径，分别保存到“fname”、“pname”中 。如果按下【取消】按钮或发生错误，则返回值都是 0。

例如，通过如下命令调用打开文件对话框来打开一个数据文件。

```
[fname,pname] = uigetfile('*.m','打开数据文件')
```

显示的对话框如图 7-38 所示，如果在对话框中选择“a1”文件，则返回值为：

```
fname =
        a1.fig
pname =
        D:\matlabR12\work\
```

注意：在得到文件名和路径后，并没有读取文件中的数据，要真正读出数据必须要在 uigetfile 函数后使用相关的文件输出和输入函数。

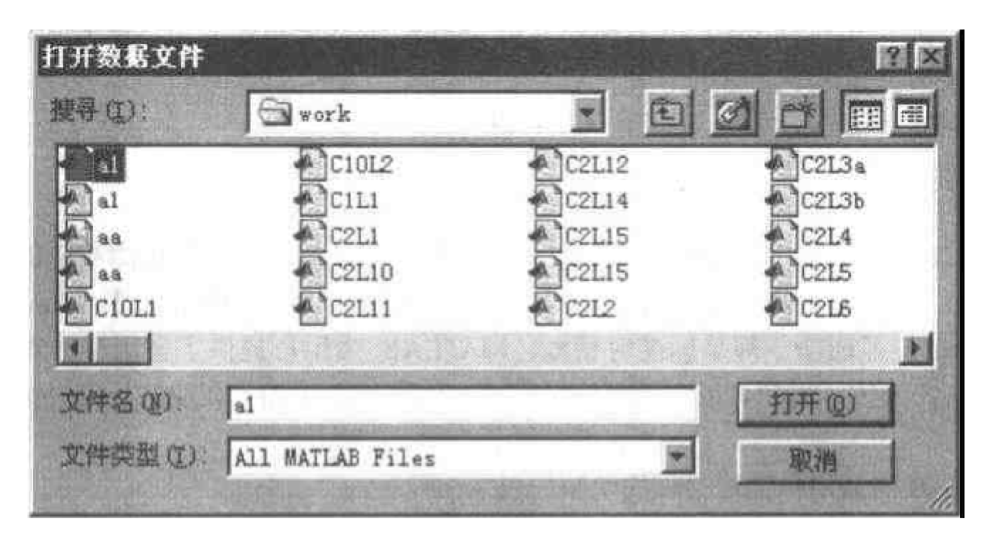


图 7-38 打开文件对话框

● uiputfile 函数

函数的调用格式如下：

◆ uiputfile 显示一个保存文件对话框，该对话框自动列出当前路径下的目录和具有缺省扩展名“.m”的文件；

◆ uiputfile('InitFile') “InitFile”指定初始时显示的文件名或文件类型，可以使用通配符“*”；

◆ uiputfile('InitFile','DialogTitle') “Dialog Title”用于指定对话框的标题；

◆ uiputfile('InitFile','DialogTitle',x,y) 指定对话框在屏幕上的位置，x、y 分别是到屏幕左边和上边的距离，单位是像素；

◆ [fname,pname] = uiputfile(...) 在按下对话框中的执行按钮后，返回选择的文

件名和路径，分别保存到“fname”和“pname”中。如果按下【取消】按钮或发生错误，则返回值是 0。

例如，通过如下命令调用保存文件对话框来保存一个数据文件。

[newfile,newpath] = uiputfile('animinit.m','Save file name')

显示的对话框如图 7-39 所示，如果在对话框中出现了所设置的保存文件名“animinit.m”，单击保存按钮保存该文件，则返回值为：

newfile =

　　animinit.m

newpath =

　　D:\matlabR12\work\

注意：函数 uigetfile 和 uiputfile 只是用来调用打开文件对话框和保存文件对话框，实际这两个对话框的作用是找到指定文件的位置，其返回值也只是文件名和路径，并没有真正打开该文件。

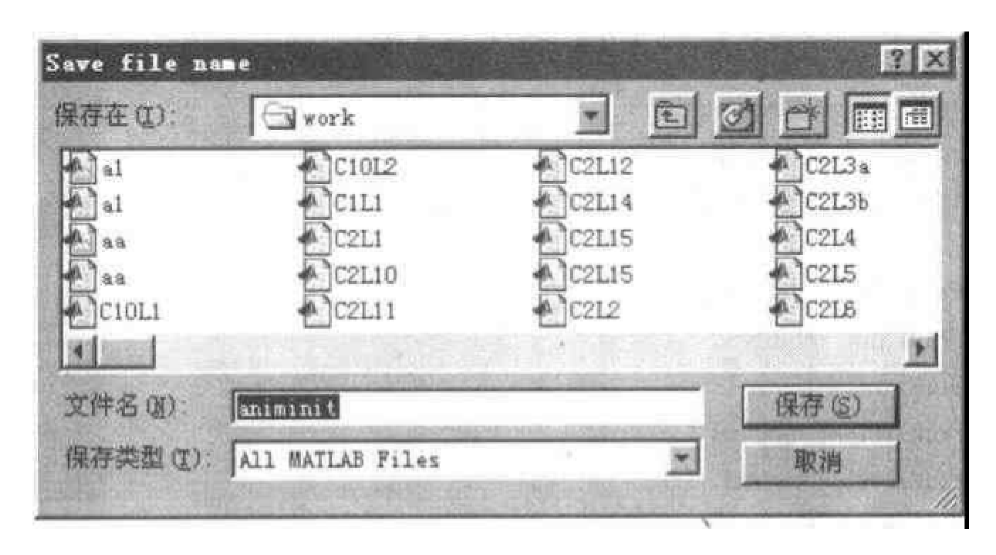


图 7-39　保存文件对话框

● uisetfont 函数

该函数用于改变文本、坐标轴或用户界面控制对象的字体属性，包括字体的种类、单位、大小、粗细和角度。函数的返回值是一个结构，包括字体的属性和相应的值。函数的调用格式如下：

◆ uisetfont　显示字体属性设置对话框，返回所选的字体属性；
◆ uisetfont(*h*)　用句柄为 *h* 的对象的字体属性值初始化对话框中的字体属性。设置后的字体属性也应用到这个对象；
◆ uisetfont(*S*)　用指定的结构“*S*”中的字体属性初始化对话框中的字体属性。结构“*S*”中必须定义了一个或多个字体属性的合法值，结构的域名和属性名必须严格相等；
◆ uisetfont(h,'DialogTitle')　用“Dialog title”指定对话框的标题。

例如，通过如下命令对文本对象的字体进行设置，命令执行结果如图 7-40 所示。

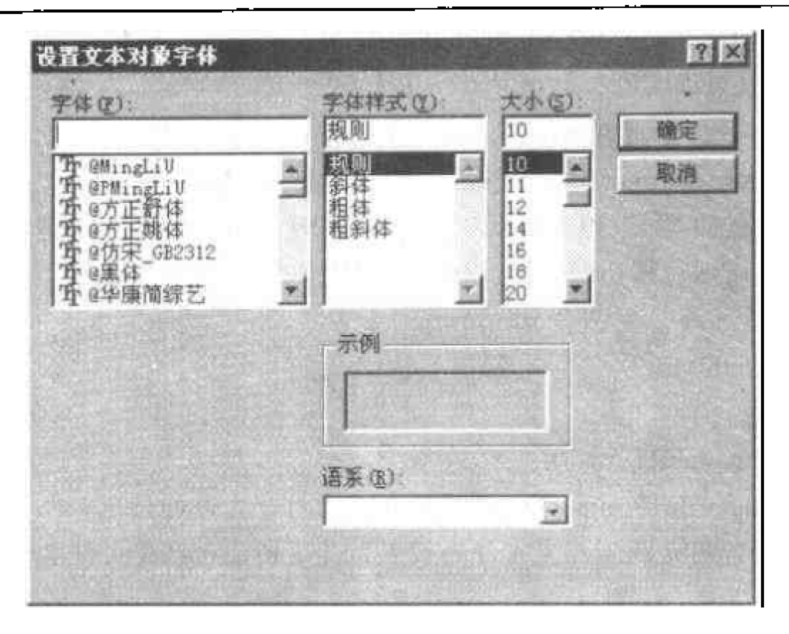


图 7-40 字体设置对话框

h = text(.5,.5,'Figure Annotation');

uisetfont(h,'设置文本对象字体')

在弹出的对话框中单击【确定】按钮，函数的返回值为：

```
ans =
      FontName: 'Helvetica'
      FontUnits: 'points'
      FontSize: 10
      FontWeight: 'normal'
      FontAngle: 'normal'
```

- uisetcolor 函数

函数的调用格式如下：

***c*=uisetcolor(handle 或 color, 'DialogTitle')**

显示一个对话框让用户选择和定义颜色。"handle" 是图形对象的句柄，该对象必须有 "color" 属性；"color" 是 RGB 颜色的三元素向量，用于指定一种缺省的颜色初始化对话框。DialogTitle 指定对话框标题。返回值 "*c*" 是所选颜色的 RGB 值。

7.6 小　　结

本章主要介绍了 MATLAB 6.0 在图形用户界面方面的强大功能，及其不同于以往的重大改进和调整。通过本章的学习，应掌握如下内容：

（1）初步了解 MATLAB 句柄图形的概念和图形对象的结构层次、理解并掌握图形对象属性的获取及其利用图形对象的属性编辑器设置对象属性的方法。

（2）掌握用户界面菜单对象（Uimenu）、用户界面上下文菜单对象（Uicontextmenu）和用户界面控制对象（Uicontrol）的两种创建方法及其属性设置。

（3）了解 GUI 设计的步骤和原则、熟练掌握如何利用 GUI 设计工具集设计用户界面菜单对象和用户界面控制对象的方法。

（4）熟练掌握用户界面对话框设计，包括专用对话框设计和标准对话框的设计步骤和方法。

第8章　Notebook 初步

MATLAB 的 Notebook 为用户提供了一个综合文字处理和科学计算的软件平台，可方便地进行科技报告、论文、著作和教材的撰写。本章从 MATLAB 6.0 的 Notebook 入手，详细介绍了在 Word 2000 文字处理系统下 Notebook 的使用方法，同时给出具体的示例，深入简出地说明如何利用 Notebook 和其他软件的结合来制作自己的电子文档和演示文稿。

Notebook 是 MathWork 公司开发的将 Microsoft Word 与 MATLAB 集成为一体、为用户营造一个融文件处理、科学计算和工程设计为一体的工作环境。它不仅有 Word 全部的文字处理功能，而且具备 MATLAB 无与伦比的数学运算能力和灵活自如的计算结果可视化能力。它既可以看作是解决各种计算问题的文字处理软件，也可看作是具备完善功能的科技应用软件。简而言之，MATLAB 的 Notebook 功能在于使用户能在 Word 环境中充分享用 MATLAB 的科技资源。

M-book 文档为用户提供了一个综合文字处理和科学计算的软件平台，不但可方便地进行科技报告、论文、著作和教材的撰写，而且用 M-book 写成的电子著作、电子讲义和网上教材不仅图文并茂，而且动静结合。那些由 MATLAB 指令构成的例题及演示，使用者都可亲自操作，举一反三，从而在“手脑并用”的环境中由此及彼、由浅入深地掌握。因此，MATLAB 的 Notebook 尤其适合于理工类学科的学习和教学，极大地方便教授者与学习者之间的交流。更为可贵的是，M-book 文档能够与 PowerPoint 和 Authorware 等应用软件相连，使计算机演讲不仅让听讲者看到事先编排的“幻灯片”和“影片”，而且可以让听讲者看到实时科学计算结果，增加听讲者的临场感和参与感，这在电子教学日益普及化的今天，有着特别现实的意义。

MATLAB 5.3 版以前的版本不支持 Word 2000，新版的 MATLAB 6.0 新增功能之一便是其 Notebook 开始支持 Word 2000。本章主要介绍 MATLAB 6.0 和 Word 2000 组合成的中文 Notebook 环境，同时兼顾 MATLAB 5.3 版本。在保证内容完整的前提下，围绕 Notebook 使用中的要点和难点展开，并强调可操作性。

8.1　Notebook 安装和启动

8.1.1　Notebook 的安装

Notebook 是 MATLAB 系统中一个相对独立的部分，MATLAB 5.x 版以后版本的 Notebook 安装是在 MATLAB 环境下进行的。其步骤如下：

（1）在 Windows 上分别安装 Microsoft Word 和 MATLAB。

（2）启动 MATLAB，打开 MATLAB 的命令窗口。

（3）在命令窗口运行命令 notebook -setup，将出现如下提示：

Welcome to the utility for setting up the MATLAB Notebook
for interfacing MATLAB to Microsoft Word
Choose your version of Microsoft Word:
[1] Microsoft Word for Windows 95 (Version 7.0)
[2] Microsoft Word 97
[3] Microsoft Word 2000
[4] Exit, making no changes

Microsoft Word Version: 3

（4）用户可按照自己的 Word 版本选择数码。本书按照 Word 2000 输入“3”，选择 Word 2000，则出现如下提示：

You will be presented with a dialog box.　Please use it to select
your copy of the Microsoft Word 2000 executable (winword.exe).
Press any key to continue...

（5）按照提示按任意键，屏幕将弹出如图 8-1 所示对话框，让用户选择 winword.exe 所在的目录，确认后，出现如下提示：

You will be presented with a dialog box.　Please use it to
select a Microsoft Word template (.dot) file in one of your
Microsoft Word template directories.　We suggest that you specify
your normal.dot file.
Press any key to continue...



图 8-1　选择 Winword 所在的路径

（6）按照提示按任意键，屏幕将弹出如图 8-2 所示对话框，让用户选择模板 normal.dot 所在的目录，确认后，出现如下提示，表示 Notebook 安装结束。

Notebook setup is complete.

注意：MATLAB 5.3 版本的 Notebook 的安装与 MATLAB 6.0 版本类似，按照提示一

步步进行即可。对于 5.0~5.2 版本，其 Notebook 的安装是 MATLAB 安装过程中的一个步骤，安装到一定阶段，屏幕会弹出一个对话框询问用户是否安装 Notebook，只要按照提示进行操作即可。

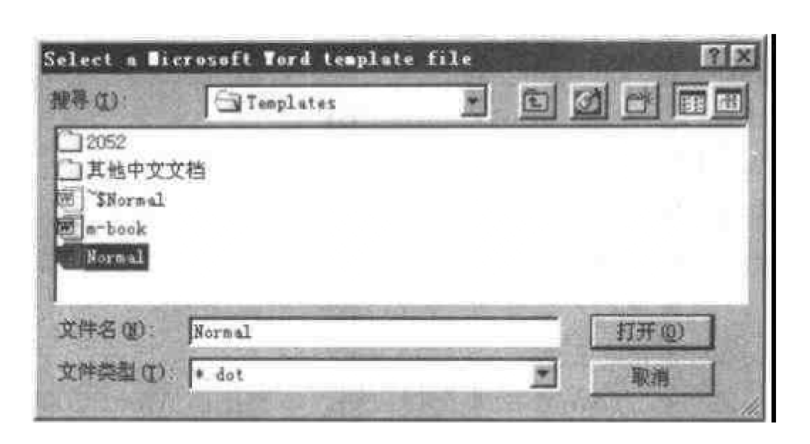


图 8-2 选择模板 Normal.dot 所在的路径

8.1.2 Notebook 的启动

启动 MATLAB 的 Notebook 有两种方法：一是从 Word 中启动；一是从 MATLAB 的命令窗口中启动。

1. 从 Word 中启动 Notebook

通常情况下，用户在 Windows 下打开 Word 后，所得到的 Word 窗口按照默认模板 Normal.dot 形式创建，在这种情况下，有三种方法从 Word 中启动 Notebook。

- 在 Word 默认窗口下（即 Normal.dot）创建 M-book。
- 选择【文件】▸【新建】命令，弹出相应对话框，选择“m-book”模板，单击【确定】按钮；此时 Word 的窗口由原来的默认式样变成“m-book”式样；如果此前 MATLAB 尚未启动，则 MATLAB 将被自动启动，用户可看到 MATLAB 的启动图标。MATLAB 启动结束后，便进入新的 M-book 文档。
- 在 Word 默认窗口下打开已有的 M-book 文件的方法与打开一般的文件一样。最常用的是选择【文件】▸【打开】命令，然后在弹出的对话框中选择要编辑的 M-book 文件。此时，系统就会自动启动 Notebook 和 MATLAB。

2. 从 MATLAB 窗口中启动 Notebook

从 MATLAB 中启动 Notebook 比较简单，只需在命令窗口中直接键入命令 Notebook。命令的调用格式如下：

- notebook　打开一个新的 M-book 文件；
- notebook PathFileName　打开一个已经存在的 M-book 文件。

PathFileName 是包括完整路径在内所需打开的 M-book 文件名。

如果 Notebook 成功启动，将产生一个如图 8-3 所示的包含 M-book 模板的 Word 界面，该界面比普通的 Word 界面多一个 Notebook 菜单，并且【文件】菜单项中多了一项【New M-book】选项。

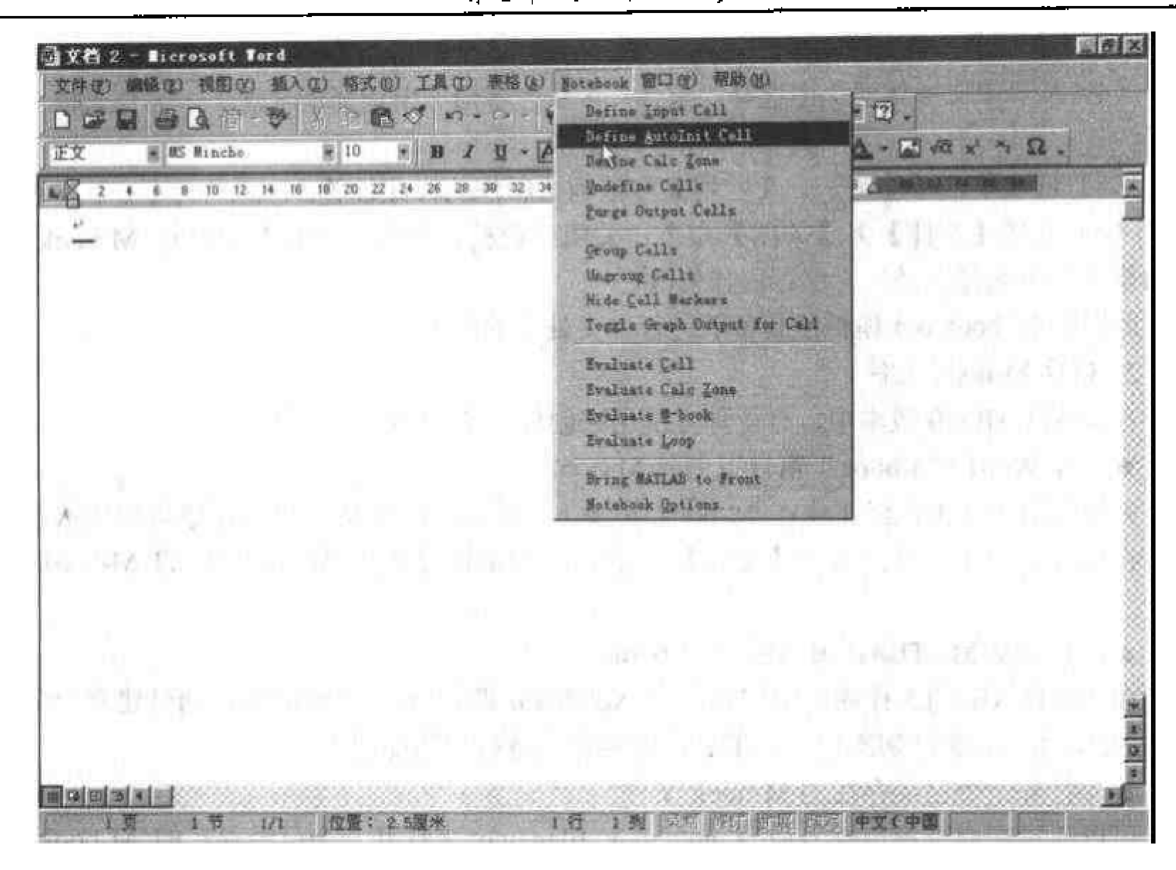


图 8-3　包含 M-book 模板的 Word 界面

8.2　M-book 模板和 Notebook 的菜单

8.2.1　M-book 模板

Notebook 的核心是 M-book 模板。它为用户提供了在 Word 环境下使用 MATLAB 的功能。该模板定义了 Word 与 MATLAB 进行通信的宏指令、文档样式和工具栏。

调用该模板时，Word 自动将 M-book 模板中定义的宏装入内存、运行 MATLAB、启动 Notebook 用户界面以及定义文本的样式和单元。同其他 Word 模板一样，用户既可以修改 Notebook 模板的原有格式，也可以加入新的格式。

1. 模板的初始配置

由于原始的 M-book.dot 模板不是专为中文环境设计，因此在初次使用时，界面的样式工具条如图 8-3 所示，即字体样式不是“宋体”，而是“MS Mingchao”（MS 明朝），此种情况下，M-book 模板尚不具备处理中文的能力，常常将汉字及某些符号显示成小方块。因此有必要先对模板进行设置，设置的步骤为：

（1）选择【格式】▸【样式】选项，打开“样式”设置对话框。

（2）单击对话框下方的【更改】按钮，弹出“更改样式”设置对话框。

（3）单击“更改样式”对话框的【格式】按钮，从弹出的菜单中选择【字体】栏，

将弹出“字体设置”对话框。

（4）从“中文字体栏”中选择“宋体”，从“字形”中选择“常规”，字号选择“五号”。

（5）依次单击【确定】、【应用】按钮确认所作的样式设置。

（6）选择【文件】▸【关闭】命令，关闭该模板，在提示“是否保存对 M-book 模板的修改”中选择“是”，保存所作修改。

此时的 M-book.dot 模板便具备了进行中文处理的能力。

2. 创建 M-book 文件

在 MATLAB 6.0 版本中，有 4 种方法用来创建一个 M-book 文件。

- 在 Word“M-book”窗口下创建 M-book

如果当前 Word 已经工作在 M-book 模板下，那么 Word 的窗口步骤将与默认状态不同，在这种情况下，直接选择【文件】▸【New M-book】选项即可创建新的 M-book 文档。

- 直接从 MATLAB 窗口创建 M-book

在 MATLAB 的工作窗口中键入命令 Notebook 即可启动 Notebook，同时建立一个新 M-book 文件。如果是初次使用，那么就需要对其进行初始配置。

- 打开一个已经存在的 M-book 文件

可以通过在命令窗口键入命令 notebook filename 来打开一个已经存在的 M-book 文件，或者像打开一个普通的 Word 文档一样，直接双击该文件。

- 将 Word 文档转换为 M-book 文件

转换的步骤如下：

（1）在 M-book 模板下，先建立一个新 M-book 文件；

（2）选择【插入】▸【文件...】选项；

（3）选择要转换的文件，单击【OK】按钮。

这样，便可将普通的 Word 文档转换为 M-book 文件。

8.2.2 Notebook 的菜单命令

M-book 模板窗口菜单栏（见图 8-3）与 Word 默认的 Normal.dot 模板的不同之处在于新增加了 Notebook 菜单，其命令项及含义如表 8-1 所示。

表 8-1 Notebook 菜单的各个子项及其含义

命令	快捷键	含义
defineinput cell	Alt+I	自定义输入单元
define autoInit cell	Alt+A	自定义自活单元
define calc zones	Alt+Z	自定义计算区
undefine cells	Alt+U	将单元转换为文本
purge output cells	Alt+P	清除输出单元
group cells	Alt+G	定义单元组
ungroup cells	Alt+p	将单元组转换为等个单元
hide cell markers	Alt+H	隐藏单元标记
show cell markers	Alt+C	显示单元标记
toggle graph output for cell	——	为每个单元锁定图形输出

续表

命令	快捷键	含义
evaluate cell	Ctrl+Enter	运行当前单元和单元组
evaluate calc zone	Alt+Enter	运行当前计算区
evaluate m-book	Alt+R	运行 M-book 中所有的单元
evaluate loop	Alt+L	循环运行单元
bring MATLAB to front	——	将 MATLAB 系统置于屏幕前
notebook options	Alt+O	定义输出显示的选项

注意：以上新增的菜单命令中，定义的快捷键与 Word 定义的某些菜单快捷键有冲突。凡是发生冲突的快捷键将按照新定义发挥作用。如要改变这种状况，用户需修改有关快捷键的命名，具体操作可参见有关介绍 Word 的书籍。

同时，在 Word 的【帮助】菜单项中增加了【About Notebook】选项，以及在【File】菜单项中增加了【New M-book】选项。

8.3 Notebook 的使用

MATLAB 的 Notebook 启动后，有关 Word 的文档便可以按照 Word 的正常方式输入文本，并对图像、表格以及数学公式等进行输入、排版和编辑。本节将重点讲述如何创建和运行 MATLAB 命令。

8.3.1 单元及单元组的基本操作

1. 输入和输出单元

在 Notebook 中，凡是参与 Word 和 MATLAB 之间进行信息交换的部分，统称为“单元（组）”（Cell or Cell Group）。由 M-book 送往 MATLAB 的命令称为“输入单元”（Input Cell）；由 MATLAB 返回到 M-book 的计算结果称为“输出单元”（Output Cell）；输入单元可以单独存在，但输出单元必须依赖输入单元而存在。

通过选择【Notebook】▸【Evaluate Cell】命令，可以激活输入单元，运行该命令，直接在其下给出计算结果，同时将结果保存在 MATLAB 的工作内存中。

对单元进行创建和运行，应该注意以下几个方面：

- 以普通文本输入的必须是 MATLAB 命令，所包含的标点符号一定要在英文状态下输入；
- 不管有多少条命令，只要可以用鼠标同时选中，即可创建或者运行；
- 如果选中命令后，按组合键 Ctrl+Enter，选中部分就成为输入单元，并在 M-book 中获得运行结果，即输出单元；
- 如果选中命令后，按组合键 Alt+D，选中部分就只是成为输入单元，并不运行，也就没有运行结果，即没有输出单元。

例如，创建一个魔术矩阵 A，直接键入“a=magic(3)”，按组合键 Ctrl+Enter，则语句变为绿色，同时在其下给出蓝色的运行结果，即输出单元。

```
a = magic(3)

a =
    8     1     6
    3     5     7
    4     9     2
```

2. 自活单元（AutoInit Cell）

自活单元（AutoInit Cell）与输入单元的惟一不同在于：用户启动一个 M-book 文件时，包含在该文件中的自活单元会自动被送去运算；而输入单元不具备这种功能。如用户需要在打开文件的同时对 MATLAB 的工作内存进行初始化工作，那么，采用自活单元特别有效。

自活单元有两种来源：一是文本形式的 MATLAB 命令，二是已经存在输入单元，只要用鼠标选中它们，然后选择【Notebook】▸【Define AutoInit Cell】命令即可。

单元和文本不同，它们是“活”的，当用户改变输入单元并再次运行时，其输出单元也会得到更新，覆盖掉原来的结果。如果用户希望保存原来的结果，就应将“活”单元转换为文本。

转换方法为：选择【插入】▸【文件…】选项，选中要转换的单元，运行【Notebook】的【Undefine Cells】选项；或者将光标置于单元中，按组合键 Alt+U。

当输入单元转换为文本后，与之相应的输入单元也被自动转换为文本。但是，如果仅仅对输出单元进行文本转换，会切断与输入单元的任何联系，不会随输入单元的变换而变化，但输入单元的性质不发生变化。此后，如再次运行输入单元，Notebook 会紧跟在输入单元之后产生一个新的输出单元。

3. 单元组（Cell Group）

单元组是由多行输入单元或自活单元组成的一个整体，它有三种创建或激活方式：

- 将输入的多行命令同时选中，选择【Notebook】▸【Define cell】（或者【Define AutoInit Cell】）命令或其快捷键，便生成单元组（或自活单元组）；
- 将输入的多行命令同时选中，单击【Evaluate Cell】命令，创建并激活单元组；
- 将已有的多个独立输入单元同时选中，单击【Group Cell】命令，便可获得一个单元组。

如被选中的多个独立单元区域内含有独立成行的文本，那么在把独立单元组合成群的同时，将原来夹在中间的文本内容移到单元组之后。

区分若干个单元是一个单元组还是一组独立的输入单元，可以选择【Notebook】▸【Show Cell Marker】命令，它将“整体”标志“[ ]”加在单元组的首尾两端，对一行行单独的输入单元，则分别显示在每一个独立单元两端。使用命令【Ungroup Cells】可以将单元组转换为一组独立的输入单元。

在如下两种情况下必须使用单元组：

- 为保证 MATLAB 命令结构（如循环、条件结构）的完整，必须使用单元组；
- 为保证输出结果（如图形）的完整，必须使用单元组。

注意：单元组不能用一行行独立的输入单元代替，否则在运行时，会显示出错信息。

同时，不管输出数值结果的 MATLAB 命令在单元组中的什么位置，单元组在输出结果时，总将数值结果放在图形结果的前面。

【例 8-1】定义单元组，生成一个完整的多变形图形（为表现单元组的特征，本例的印刷格式即为单元组的显示格式，不予改变），其效果如图 8-4 所示。

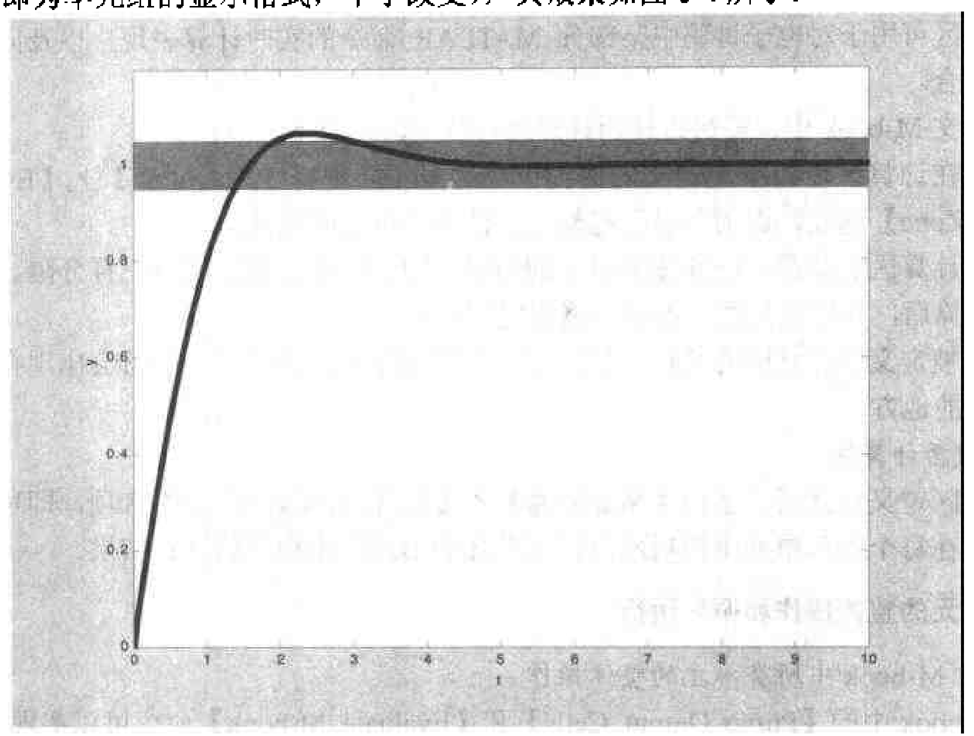


图 8-4　单元组产生的完整图形

```
t=0:0.1:10;y=1-cos(t).*exp(-t);
tt=[0,10,10,0];
yy=[0.95,0.95,1.05,1.05];
fill(tt,yy,'g'),axis([0,10,0,1.2,5]),xlabel('t',5),ylabel('y',5)
hold on
plot(t,y,'k','LineWidth',4)
hold off
ymax=max(y)

ymax =
     1.0669                    %显示计算的数值结果，为蓝色
```

8.3.2　计算区的基本操作

计算区（Calc Zone）是一个由文本和单元组（包括输入和输出）组成的连续区，用于描述某个具体的作业和问题。Notebook 将计算区定义为 Word 的一个“节”，在计算区内的计算操作与电子文档的其他部分独立，不管计算区内含有多少输入单元群，只需要一次操作就可以实现对区内所有单元的计算，并将所有的生成变量存放在同一个 MATLAB 的工作内存中。

1. 创建计算区

创建计算区的方法是：先选定包含输入输出单元和文本在内的一个连续区，然后选择【Notebook】▸【Define Calc Zone】选项。此时，整个计算区的两端被一对“整体”标志“[ ]”括起，表示计算区被定义成功。

计算区可用于对电子讲稿中某部分 MATLAB 命令的实时计算表现，以及向学生布置作业等场合。

在中文 M-book 中，要慎重使用计算区，特别注意以下几点：

- 在计算区没有事先定义的情况下，尽量不要选择【Notebook】▸【EvaluateCalc Zone】选项，否则容易引起整个文档输出单元的混乱。
- 计算区运算前，应使输入单元的结束与随后普通文档之间有空行分隔。否则在运算后，有可能使原来的文档格式发生变化。
- 被定义的计算区两端最好是空行，否则所定义计算区的分隔符会出现在意想不到的地方。

2. 激活计算区

计算区定义好之后，选择【Notebook】▸【Evaluate Calc Zone】选项，即可激活计算区，将会在每个输入单元组的后面以输出单元的形式给出相应的计算结果。

8.3.3 单元的整体操作和循环运行

1. 对 M-book 中所有单元的整体操作

Notebook 中的【Purge Output Cells】和【Evaluate M-book】命令可以实现对 M-book 文件中所有单元进行整体操作。

【Evaluate M-book】表示运行整个 M-book 文件，即将文档中所有单元送入 MATLAB 中运行。不管光标处在文档的何处，运行总是从文件的首部开始。运行时，不仅对所有的原输出单元中的内容更新，而且会补写新的输出单元。这在保证整个 M-book 文件中所有命令、数据和图形的一致性方面有重要的作用。

注意：如果 M-book 文档的输出单元没有经历过编辑（如图形的“对中”、输出细胞的挪动以及其前后空行的删除等）操作，使用【Evaluate M-book】命令可得到良好的预期效果，否则有可能造成整个版面的混乱。因此，应用该命令时要慎重，尤其对较大的 M-book 文件而言，更是如此。

而【Purge Output Cells】是删除 M-book 文件中所有的输出单元。首先选中整个文件，然后运行该命令，所有的输出单元将会被删去。该命令在撰写报告和布置作业时常常用到。如果希望删除部分的输出单元，只需选中要删除单元所在的区域。

2. 单元的循环操作

Notebook 提供有循环运行单元的命令【Evaluate Loop】，其实现循环运行的操作步骤如下：

（1）选择要重复运行的输入单元，可以包含合法的文字或输出单元。

（2）选择【Evaluate Loop】命令，弹出如图 8-5 所示的对话框。

（3）在【Stop After】栏中键入所需重复运行的次数，然后单击【Start】按钮。Notebook 开始执行命令，并显示运行的次数。

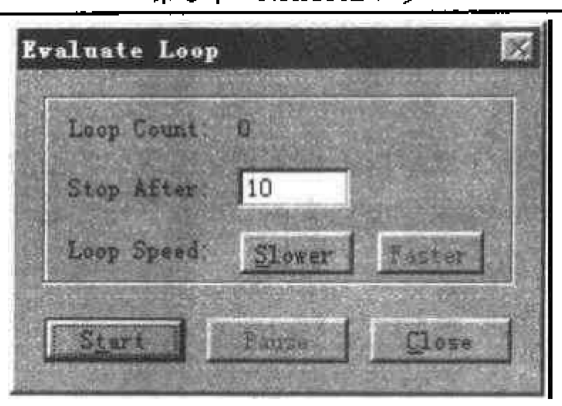


图 8-5 循环对话框

（4）如需要在每一个循环后加入延迟，可单击【Slower】按钮，反之单击【Faster】按钮，单击【Pause】按钮表示暂停运行。

（5）单击【Close】按钮，停止运行命令。

【例 8-2】利用【Evaluate Loop】菜单选项绘制图形。

（1）先运行以下命令：

```
clear;x=0:10;k=1;hold on
```

（2）选亮以下三行指令，打开循环运行对话框，并取重复次数为 10，单击【Start】按钮，就可看到每循环一次绘制一条直线的全过程。

```
y=k*x;
plot(x,y)
k=k+1;
```

（3）待循环结束，单击【Close】按钮，关闭对话框。在以上运作结束后，最好再运行“hold off”指令，避免以后图形被重叠。绘制的图形如图 8-6 所示。

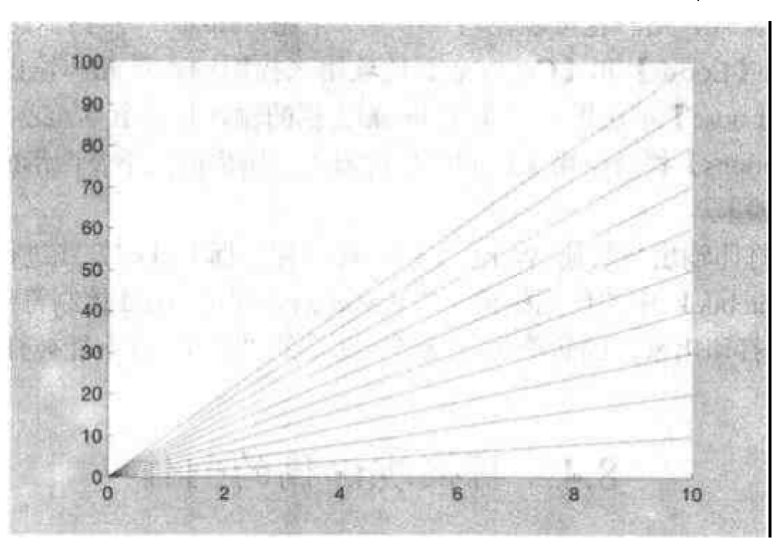


图 8-6 由【Evaluate Loop】菜单选项所绘制的图形

8.3.4 输出与文档打印

1. 输出格式控制

Notebook 的输出单元包含 MATLAB 的各种输出结果，包括数据、图形和错误信息等。输出数据的格式通过【Notebook Options】命令进行控制，运行该命令后，出现如图 8-7 所示对话框。

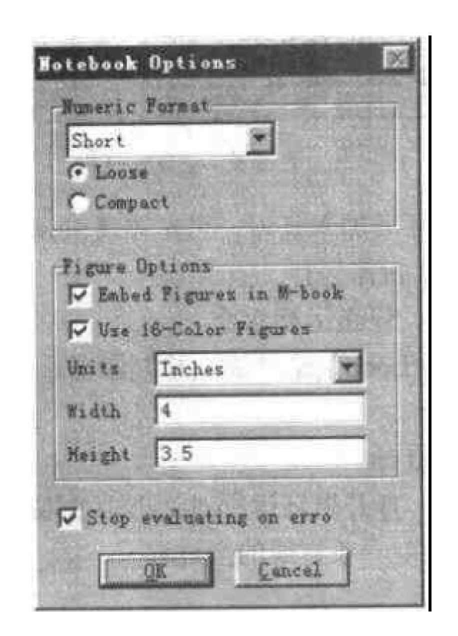


图 8-7 控制输出单元格式的对话框

【Numeric Format】选项组中的下拉式列表中有 8 种可选的格式：Short、Long、Hex、Bank、Plus、Short e、Long e 和 Rational。它们的作用和 MATLAB 工作窗口中的 format 命令完全相同，同样，用户也可以在输入单元中采用 format 命令进行设置。

对话框中的【Loose】和【Compact】选项用来控制输入单元和输出单元之间的空白区间。如选择【Loose】单选框，则在 M-book 文档的输入和输出单元之间加入一个空行。

【Figure Options】栏目组用于控制图形的嵌入、图形的大小及图形的颜色。

2. 文档的打印

M-book 的打印输出与其他 Word 文档一样，按照标准打印方式进行，但是，需要特别指出的是：Notebook 中的单元标记（Cell Marker）属于 Word 控制符号，只能在屏幕上显示，并不能被打印出来。因此单元组或单元两端的“[ ]”并不能被打印出来。

8.4 科技演讲稿的制作

随着大量的学术报告会和高等教学的日益现代化，有越来越多的演讲稿和教学讲义采用电子设备进行制作，特别是那些需要在现场进行复杂的数学计算或者改变参数进行实时计算，并给出数值和图形结果的场合，MATLAB 的 Notebook 和一些专门软件结合制作的稿件便显示了其强大的功能。

本节将以常用的制作演讲稿的软件 PowerPoint 为例，介绍如何制作科技演讲稿。

由于用于超级链接的 Word 文档是用 M-book 模板制作的，因此，这些 Word 文档的开启也就意味着“进入了 MATLAB 的 Notebook 环境”。并且这种 Notebook 环境可以执行和实现 MATLAB 的绝大部分计算功能。也就是说，此时的演示稿将 Word、PowerPoint 和 MATLAB 有机地紧密联系在一起。

使用 PowerPoint 制作演示文稿的步骤如下：

（1）先制作普通的演示文稿。

启动 PowerPoint，进入 PowerPoint 的工作环境：选择【文件】的【新建】命令，在打开的对话框中选择“空演示文稿”，再在随后打开的对话框中选择自动版式，例如，选择第一排第二个版式，得到如图 8-8 所示的母片。

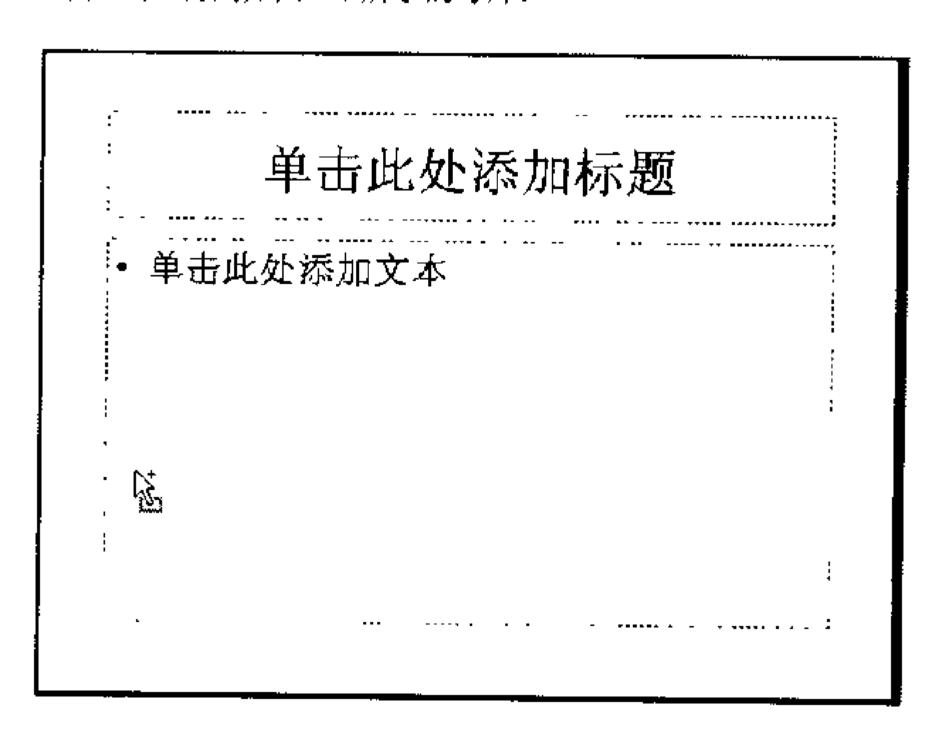


图 8-8　第一张母片的版式

在母片的上方栏中键入“MATLAB 6.0 工程教学应用”，在下方栏中键入“电子文稿演示”；选择【插入】▸【新幻灯片】选项，便可引出第二张母片，写入各个章节的内容，依次类推。

选择【格式】▸【应用设计模板】选项，为幻灯片选择合适的背景。

（2）将有 M-book 的文档以超级链接的内容插入幻灯片。

将多个幻灯片链接为一个整体：为幻灯片中的“对象”（如文字和图形等）设计链接到本幻灯片文件的其他幻灯片的“超级链接”。例如，选亮文字对象“MATLAB 6.0 工程教学应用”，然后，选择【插入】▸【超级链接】选项，在弹出的对话框中，按提示在第二栏填写被链接的幻灯片（该幻灯片已存在）。此时，做好超级链接的对象下将被划上下划线，如图 8-9 所示。

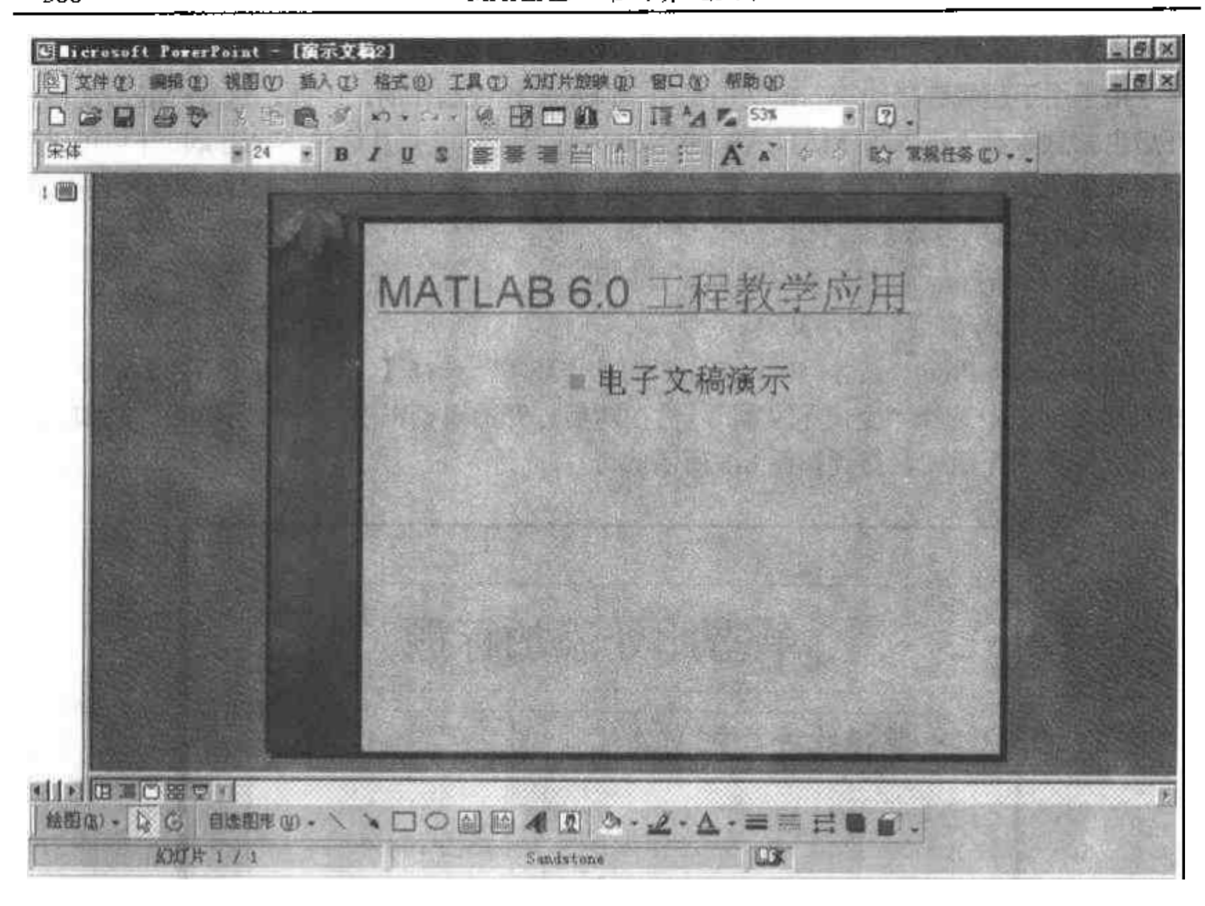


图 8-9 做好超级链接的幻灯片

为幻灯片中“对象”设计链接到文档的“超级链接”。在对话框的第一栏中填写用户要链接的 M-book 文件。

如果要链接多个文件，按照以上步骤重复执行。

此时，用户所需内容的演示文稿便已完成。

8.5 小结

MATLAB 的 Notebook 是一个综合文字处理和科学计算的软件平台，可方便地用于科技报告、论文、著作和讲义教材的撰写。本章主要讲述了在 Word 2000 文字处理系统下的 Notebook 的使用方法，同时结合相关软件介绍了电子演示文稿的制作。通过本章的学习，应该掌握如下内容：

（1）了解并初步掌握 Notebook 在 Word 97 和 Word 2000 下的安装过程和启动方法。

（2）了解并掌握 Notebook 的 M 模板以及 Notebook 的菜单命令。

（3）熟练掌握 Notebook 单元组和计算区的基本操作，了解并掌握单元的整体操作和循环运行，以及文档的输出与打印。

（4）掌握利用 Notebook 和相关软件相结合，进行科技演讲稿的制作的方法。

习 题

1．按照本章所述安装 Notebook，并采用不同的方法启动 Notebook。

2．尝试、熟悉并掌握 M-模板的 Notebook 菜单项，及其各个子菜单的功能。

3．在 Notebook 环境下，绘制函数 $y=\dfrac{\cos(5x+2)}{\sin(3x+1)}$ 的图像，并求解当 x=2 时的函数值。

第 9 章　综合实例

本章给出工程和数理学科方面的几个实例，说明如何用编程的方法进行这一类问题的解决。相信通过本章的学习，既可以对已经掌握的内容作较为全面的复习，又可以对 MATLAB 在实践中的具体应用有更深刻的认识。

9.1　振动问题

9.1.1　单自由度体系有阻尼自由振动

运用 MATLAB 求解这个问题时，一般要经历建模和编程两个过程，只有在建模正确的前提下，方能得出正确的结果。

1. 建立计算模型

由动力学可知，单自由度体系有阻尼自由振动的振动方程为：

$$m x + c x + kx = 0$$

可以转化为：

$$x + \xi\omega x + \omega_n{}^2 x = 0$$

其中，$\omega_n = \sqrt{\dfrac{k}{m}}$，$\xi = \dfrac{c}{2\sqrt{mk}}$，那么运动方程的解为：

$$x(t) = Ae^{-\xi\omega_n t}\sin(\omega_d t + \phi)$$

其中，$A = \sqrt{\dfrac{(v_0 + \xi\omega_0 x_0)^2 + (x_0\omega_d)^2}{\omega_d{}^2}}$，　$\phi = \mathrm{tg}^{-1}\left(\dfrac{x_0\omega_d}{v_0 + \xi\omega_n x_0}\right)$，

$$\omega_d = \omega_n\sqrt{1-\xi^2}$$

x_0 表示初始位置，v_0 表示初始速度。

现在，分别设 ξ 从 0.1 到 1，公共参数 $\omega_n = 10$，$x_0 = 1$，$v_0 = 0$，计算的终止时间 $tf=2$。试求运动方程的解，并画出波形。

2. MATLAB 编制解算程序

编写 M 文件 C11L1.m 如下：

```
%首先清空 MATLAB 的工作空间
clear;
%给定初值
wn=10;
tf=2;
x0=1;
v0=0;
%计算不同的ξ值所对应的振型
for j=1:10;
    eta(j)=0.1*j;
    wd(j)=wn*sqrt(1-eta(j)^2);
%求振幅 A
    a=sqrt((wn*x0*eta(j)+v0)^2+(x0*wd(j))^2)/ wd(j);
%为了求四象限相位角调用函数 atan2
    phi=atan2(wd(j)*x0,v0+eta(j)*wn*x0);
%设定自变量数组 t
    t=0:tf/1000:tf;
%求过渡过程
    x(j,:)=a*exp(-eta(j)*wn*t).*sin(wd(j)*t+phi);
end
%在同一个图形窗口中绘制不同的ξ值所对应的振型
plot(t,x(1,:),t,x(2,:),t,x(3,:),t,x(4,:),...
    t,x(5,:),t,x(6,:),t,x(7,:),t,x(8,:),...
    t,x(9,:),t,x(10,:))
grid on
%新建一个图形窗口，绘制三维网格图
figure
mesh(x)
>>
```

程序的运行结果如图 9-1 和图 9-2 所示，曲线反映出不同的 ξ 值对固有振动模态的影响。图 9-2 是其三维图。

从三维图中可以形象地看出 ξ 对固有振型的影响，按下图形窗口工具栏上的【Rotate 3D】按钮进行三维旋转，以得到更为清晰直观的概念。

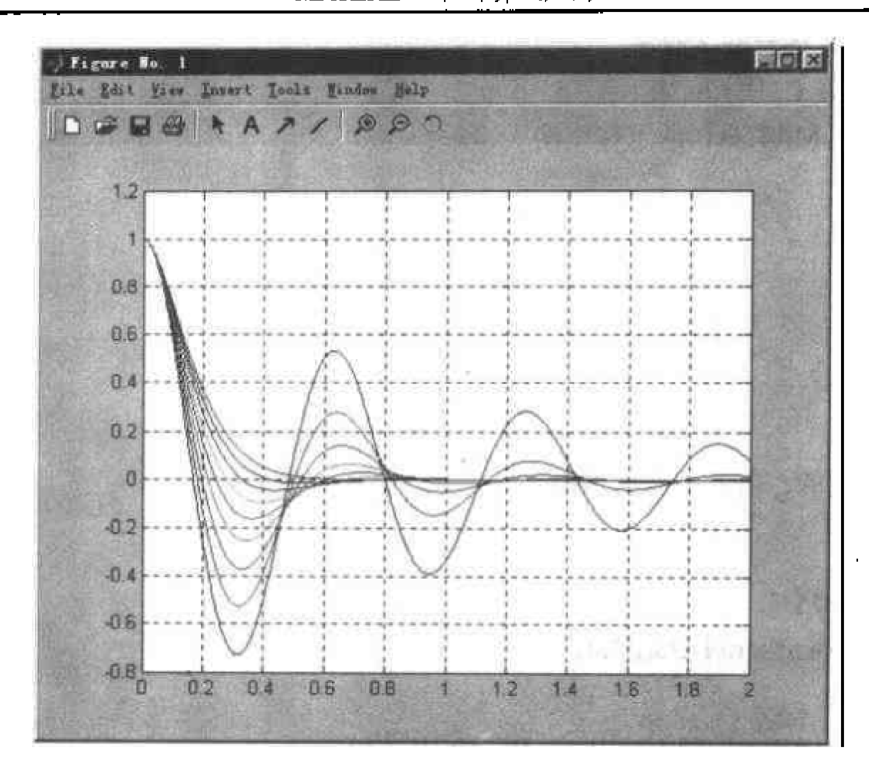


图 9-1 不同的 ξ 值的固有振型

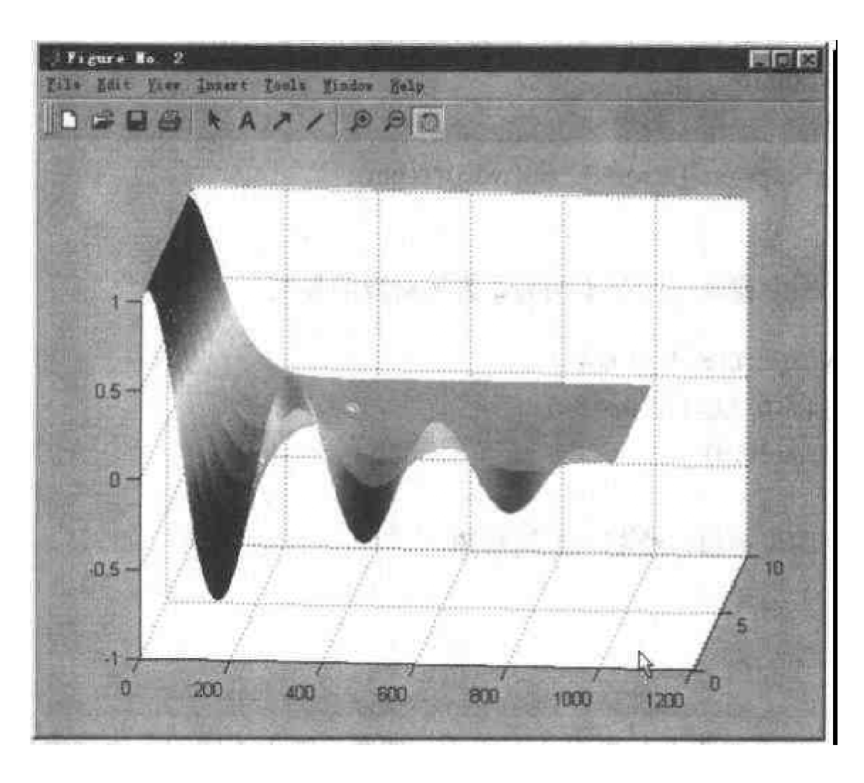


图 9-2 不同 ξ 值的固有振型三维网格图

如果改变初始条件令 $x_0=0$，$v_0=1$，即给定一个初始速度，其运动曲线实际上就是系统的脉冲过渡函数，如图 9-3 和图 9-4 所示。由于脉冲函数的幅度无穷大，而持续时间

却是无穷小，其面积是一个单位，因此，脉冲激励的最后效果是：可在 t=+eps 处形成一个单位的初速度 v_0，由它产生的波形就是脉冲过渡函数。

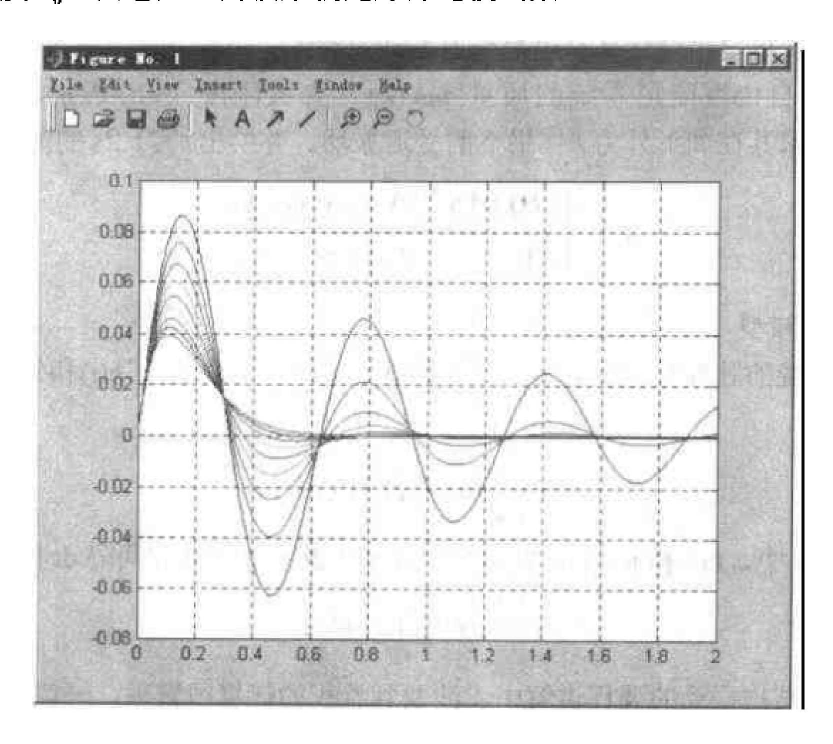


图 9-3 脉冲函数不同的 ξ 值的固有振型

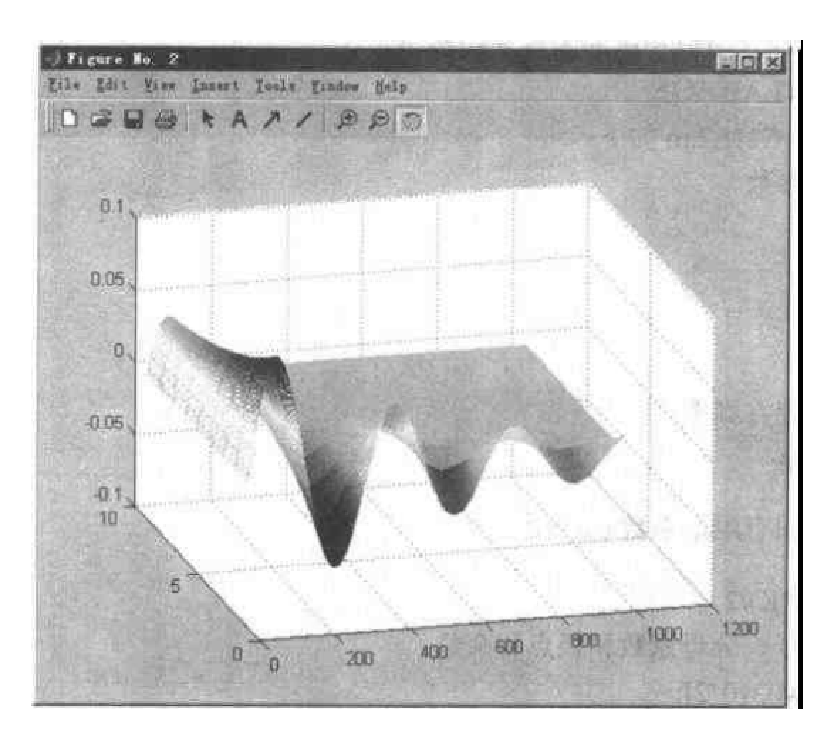


图 9-4 脉冲函数不同 ξ 值的固有振型三维网格图

9.1.2 单自由度体系有阻尼受迫振动

下面讲述如何从原始方程出发，不经过手工推导而依靠 MATLAB 直接求解的方法。并举例为证，求解单自由度体系有阻尼受迫振动问题。

例如，设单自由度阻尼系统的质量 m=1kg，弹簧刚度 k=100N/m，速度阻尼系数 $c = 4N \cdot s/m$，求其在如下外力 f 作用下的受迫振动，并绘出 $t \le 1.2$s 的波形。

$$f = \begin{cases} t/0.015 & 0 \le t \le 0.15\text{s} \\ 10 & 0.15 \le t \le 1.2\text{s} \end{cases}$$

1. 建立计算模型

首先求出系统的脉冲过渡函数 $h(t)$，则受迫振动的波形就等于 $h(t)$和外加力 $f(t)$作卷积的结果。

$$x(t) = \int_0^t h(t-\tau) \cdot f(\tau) \mathrm{d}\tau$$

在 MATLAB 中，$h(t)$和 $f(t)$都可以用数组 h 和 f 表示，其取样间隔 dt 应相同，便有

$$x = \text{conv}(h, f) * \mathrm{d}t$$

在动力学及结构振动的课程讲解中，涉及到卷积的计算很繁琐，一般课堂上只讨论一些标准的有解析表达式的激励信号，如方波和正弦波等，如果利用 MATLAB，就不必受到繁琐计算的限制。例如在本例中，利用函数 residue 求得脉冲过渡函数，然后采用卷积函数 conv，根据输入函数和脉冲函数求解输出。

2. 编制 MATLAB 程序

编写 M 文件 C11L2.m 如下：

```
%给定初始参数
m=1;
c=4;
K=100;
dt=0.015;
%求系统的固有频率
w0=sqrt(K/m);
%求系统的固有阻尼系数 eta=ξ
eta=c/sqrt(m*K)/2
%分别求分子、分母系数和极点和余数
a=[1,2*eta*w0,w0^2];
b=1;
[r,p]=residue(b,a);
t=0:dt:1.2;
%求系统的脉冲响应
```

```
h=r(1)*exp(p(1)*t)+r(2)*exp(p(2)*t);
%给出外加力 f 的采样值
f=[1:10,10*ones(1,70)];
%将脉冲响应 h 和外加力 f 作卷积
x=conv(h,f)*dt;
%绘制图形，并划分网格
plot(t(1:80),x(1:80))
grid
%对响应求导，得出速度
v1=diff(x)/dt;
%每隔 5 个点列出结果
[t(1:5:80)',f(1:5:80)',x(1:5:80)',[0,v1(6:5:79)]']
```

程序的运行结果如下，得出如图 9-5 所示的响应图。

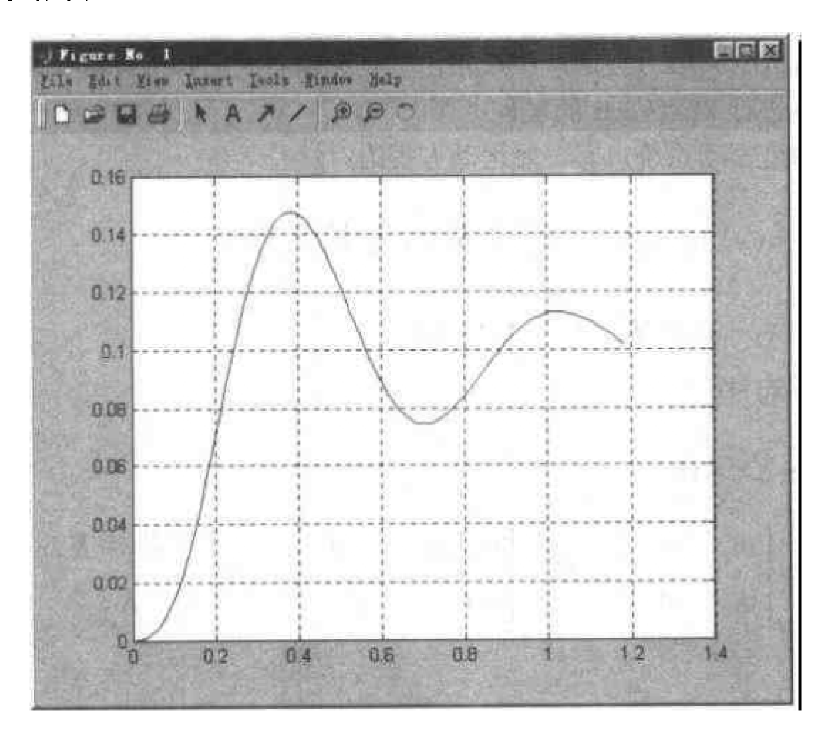


图 9-5　单自由度阻尼体系的受迫振动

```
eta =
    0.2000
ans =
         0    1.0000         0         0
    0.0750    6.0000    0.0069    0.2571
    0.1500   10.0000    0.0372    0.6116
    0.2250   10.0000    0.0869    0.6606
```

```
    0.3000      10.0000      0.1297       0.3907
    0.3750      10.0000      0.1476       0.0096
    0.4500      10.0000      0.1386      -0.2772
    0.5250      10.0000      0.1140      -0.3611
    0.6000      10.0000      0.0892      -0.2559
    0.6750      10.0000      0.0757      -0.0592
    0.7500      10.0000      0.0769       0.1139
    0.8250      10.0000      0.0884       0.1893
    0.9000      10.0000      0.1022       0.1575
    0.9750      10.0000      0.1113       0.0608
    1.0500      10.0000      0.1127      -0.0390
    1.1250      10.0000      0.1078      -0.0948
>>
```

9.1.3 双自由度可解耦系统的振型分析

利用 MATLAB 可以方便地解决双自由度可解耦系统的振型分析。同样采用先建立数学模型，而后再进行 MATLAB 的编程工作。

对于双自由度振动系统，其一般运动方程为：

$$m_1 x_1+(c_1+c_2)x_1-c_2 x_2+(K_1+K_2)x_1-K_2x_2=0$$

$$m_2 x_2+c_2(x_2-x_1)+K_2(x_2-x_1)=0$$

写出矩阵形式为：

$$M x+C x+Kx=0$$

其中，$M=\begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix}$，$C=\begin{bmatrix} c_1+c_2 & -c_2 \\ -c_2 & c_2 \end{bmatrix}$，$K=\begin{bmatrix} K_1+K_2 & -K_2 \\ -K_2 & K_2 \end{bmatrix}$，

$X=[x_1,\ x_2]^T$。

如果设 $C=0$，即无阻尼情况，则系统可以解耦为两种独立的振型。

1. 建立计算模型

该方程可采用如下步骤进行求解：

首先用 eig 函数求解矩阵 $K-\lambda M$ 的特征值 L 和特征向量 U，U 和 L 满足：

$$U'*M*U=I$$

$$L=U'*K*U=\begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

在原始方程 $M\ddot{x}+Kx=0$ 两端各左乘 U，在中间乘以对角矩阵 $U'*U$，得：

$U'*M*U'*U*\ddot{x}+U'*K*U'*U*x=0$

作变量置换 $z=\begin{bmatrix} z_1 \\ z_2 \end{bmatrix}=U'*\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}=U'*x$，得：$\ddot{z}+L*z=0$

这是一个对角矩阵方程，可以把它分为两个方程：

$\ddot{z}_1+\lambda_1 z_1=0$

$\ddot{z}_2+\lambda_2 z_2=0$

这就意味着两种振型可以解耦，令 $\omega_i^{\ 2}=\lambda_i$，ω_i 为第 i 个振型的固有频率（i=1，2）。

在上述的解耦振型中，给出初始条件 x_0，x_{d0}，化为 z_0，z_{d0}，即可分别求出其分量 z_1，z_{d0}，再按照 $x=\mathrm{inv}(U')*z$ 变换为 x。

设位置和速度的初始条件分别为再按照 $x=\mathrm{inv}(U')*z$ 变换为 x。

设位置和速度的初始条件分别为 $x_0=[x_{01},x_{02}]^T$，$x_{d0}=[x_{d01},x_{d02}]^T$，则这三个步骤得到的最后公式为：

$$x(t)=\sum_{i=1}^{2}[u_i]\cdot([u_i]^T Mx_0\cos\omega_i t+\frac{1}{\omega_i}[u_i]^T Mx_{d0}\sin\omega_i t$$

2. MATLAB 编程

按照建立计算模型的步骤编写 M 文件 C11L3 如下：

```
%输入各原始参数
m1=1;
m2=9;
K1=4;
K2=2;
%输入初始条件
x0=[1;0];
xd0=[0;-1];
tf=20;
%组成参数矩阵
M=[m1,0;0,m2];
```

```
K=[K1+K2,-K2;-K2,K2];
%求广义特征向量和特征值 U，L
[u,l]=eig(K,M);
%设置时间间隔采样点，并将输出变量 x 初始化
t=linspace(0,tf,101);
x=zeros(2,101);
%分别处理两个特征值
for s=1:2
    %解耦后的质量 alfa
    alfa=sqrt(u(:,s)'*M*u(:,s));
%将特征向量[ui]归一化
    u(:,s)=u(:,s)/alfa;
    d=diag(l);
%分别求对应于两个特征值的分量
    w=sqrt(d);
    for j=1:2
    xt=u(:,j)*(u(:,j)'*M*x0*cos(w(j)*t)+u(:,j)'*M*xd0/w(j)*sin(w(j)*t));
%将两个分量累加
    x=x+xt;
end
end
```

分别绘制两个振型图：

```
for r=1:2
    subplot(2,1,r)
    plot(t,x(r,:))
    grid
    xlabel('时间，秒');
    ylabel(['响应 x',num2str(r)]);
end
```

程序的运行结果如图 9-6 所示。

本例是对理想的可解耦情况采用传统的分析方法来获得解析的结果。其实利用 MATLAB 工具，无需设置 $C=0$，也无需解耦，就可用很简洁的程序方便地求出运动方程的数值解。其基本思路为：

将原始运动方程化为典型的四阶状态方程：

$$y = A * y$$

其中，y=[$x1$;$x2$;$xd1$;$xd2$]，$xd1$,$xd2$ 分别为 $x1$,$x2$ 的导数。

$$A=\begin{bmatrix}0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\\ -M/K & & & -M/C\end{bmatrix}$$

这样，问题就直接转变为对微分方程的求解，请读者思考，自行上机练习。

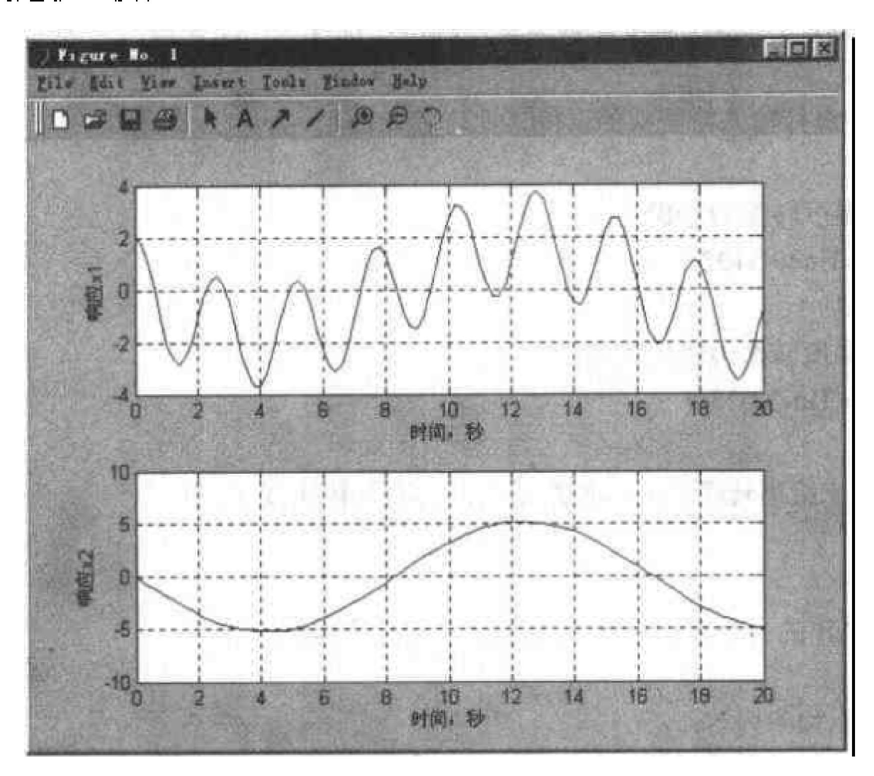


图 9-6　双自由度体系的振动的输出波形

9.2　热力学和分子问题

9.2.1　温度的转换

在实际应用的过程中，经常要遇到单位转换的问题，本节将给出一段程序，用于将用户输入的摄氏温度转为华氏温度，将华氏温度转换为摄氏温度。

1. 计算模型的建立

在摄氏温度和华氏温度之间转换的公式为：

$$T_{摄氏温度}=5\times(T_{华氏温度}-32)/9$$

即：

摄氏温度转换为华氏温度　　$T_{out}=\frac{9}{5}T_{in}+32$

华氏温度转换为摄氏温度 $T_{out}=\frac{5}{9}(T_{in}-32)$

因此，在程序中要先考虑由用户选择的转换方向，来确定需要选用的公式。

2. MATLAB 程序的编写

根据步骤 1 的分析，编写 M 文件 C11L4 如下：

```
k=input('选择 1：摄氏温度转换为华氏温度；选择 2：华氏温度转换为摄氏温度；请键入数字：');
Tin=input('选择输入待变换的温度（允许输入数组）：');
if k==1
  %摄氏温度转换为华氏温度
    Tout=Tin*9/5+32
else if k==2
  %华氏温度转换为摄氏温度
    Tout=(Tin-32)*5/9
else
    disp('未给出转换方向，转换无效；请输入转换数字：')
end
end
```

在 MATLAB 的命令窗口中键入 C11L4，运行该程序：

C11L4

屏幕将显示如下内容：

“选择 1：摄氏温度转换为华氏温度；选择 2：华氏温度转换为摄氏温度；请键入数字：”

选择数字 1，按 Enter 键，屏幕将显示：

“选择输入待变换的温度（允许输入数组）：”

键入数字 20，求 20℃所对应的华氏温度，屏幕上将给出：

```
Tout =
      68
```

当然，这是个很简单的例子，但是可以看出，如果需要重复地进行温度转换，这种方法很麻烦。下面就讲述如何建立一个用于温度转换的图形用户界面，只要用户键入需转换的温度，立刻就会有结果在界面上显示出来。

9.2.2 设计一个温度转换的图形用户界面

1. GUI 界面要实现的功能

要设计一个用于温度转换的图形用户界面，要求在该图形用户界面中可以实现下述功能：

在左边的文本框中输入华氏温度，单击 Enter 键后，在右边的文本框中出现相应的摄氏温度；在右边的文本框中输入摄氏温度，单击 Enter 键后，在左边的文本框中出现相应的华氏温度。同时，中间的温度显示条可以显示出对应的温度。

在中间温度显示条所在的区域单击鼠标左键，显示条显示鼠标位置对应的温度，同时在两边的文本框中给出相应的华氏和摄氏温度。

2. 所需控制对象的类型

为了实现上述功能，在图形用户界面中需要如下 GUI 控制对象：

- 两个坐标轴（Axis）对象，分别显示华氏温度和摄氏温度的大小；
- 两个静态文本框（Text），用于显示华氏温度和摄氏温度；
- 两个可编辑文本框（Edit），用来直接输入已知的温度；
- 两个静态文本框（Text），一个的前景色设为红色，用来显示温度的高低；一个设置为白色，用作温度显示条的背景以及接受鼠标的位置。

3. 控制对象的添加和安置

控制对象的添加步骤如下：

（1）首先在 MATLAB 命令窗口中键入【Guide】命令，打开图形用户界面设计工具集。

（2）单击控制对象工具栏的【Static Text】按钮，将在图形区产生一个静态文本框的控制对象。用鼠标拖动该对象，设置合适的大小和位置。用相同的方法建立第二个静态文本框。将两个静态文本框分置两边，用于输入华氏温度和摄氏温度。

（3）单击【Axis】按钮，在图形区建立坐标轴对象，用同样的方法建立第二个。

（4）单击【Edit Text】按钮，在图形区建立可编辑文本框对象，用同样的方法建立第二个。

（5）单击【Static Text】按钮，建立用于显示温度高低的显示条的静态文本框。

此时，图形区内设置好具有各种原始控制对象的图形用户界面，如图 9-7 所示。

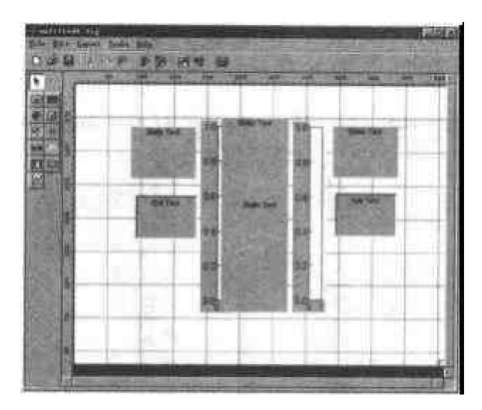


图 9-7　设好原始控制对象的图形用户界面

注意：中间的长条形静态文本框有两个，目前重叠在一起，由于颜色相同，在没有设置属性之前，只能看出两个“Static Text”的标记。

提示：如果有控制对象被另一个挡住，可以选中要显示在前面的对象，单击鼠标右键，在弹出的上下文菜单中单击【Bring to Front】命令，反之单击【Send to Back】命令。

为美观起见，两个可编辑文本框大小最好一致。可以在建立好一个控制对象后，单击【Edit】▸【Copy】命令，然后单击【Paste】命令；或者单击鼠标右键，在弹出的下拉菜单中，单击【Duplicate】命令，复制一个相同的控制对象。

4. 设置属性

设计好如图 9-7 所示的图形用户界面，如果想使界面发挥其应有的功能，还需要对各个控制对象的属性进行设置。设置的步骤如下：

（1）单击工具栏上的【Property Inspector】按钮，打开属性检查器窗口。

（2）选中左上方的静态文本框，在属性检查器窗口中设置各项属性。单击左侧属性列表中的“String”，在右边的编辑栏中键入：

“华氏温度

Dgrees

Fahrenheit”

选中 Size 属性，设置字体大小为 12。用同样的方法设置右侧的静态文本框，用于显示摄氏温度。

（3）选中左边的坐标轴对象，单击 Ylim 属性旁边的“+”，打开其子属性 x、y，键入 y 轴的刻度范围为-40~240，在 Ytick 和 YtickLabel 属性中键入 y 轴的刻度值[-40.0,-20.0,0.0,20.0,40.0,60.0,80.0,100.0,120.0,140.0,160.0,180.0,200.0,220.0,240.0]；在 YaxisLocation 属性中设置 y 轴标注的位置是在左侧还是在右侧，选择“left”；选中 XtickLabel 属性，给 x 轴标注华氏字样。同理，选中右边的坐标轴对象，作同样的修改，但右边坐标轴的 YaxisLocation 属性应该设置为“right”。

（4）选中用于显示温度高低的静态文本框，将显示条的“Backgroudcolor”（背景颜色）设置为红色，将用于接受鼠标位置的文本框的“Backgroudcolor”（背景颜色）设置为白色。

（5）取消坐标轴对象、可编辑文本框对象和用于显示温度高低的静态文本框的 String 属性。

此时，设置的图形用户界面如图 9-8 所示。

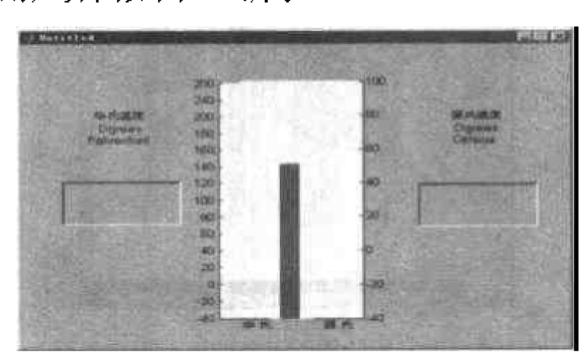


图 9-8　初步设置属性的图形用户界面

5. 编写 Callback 程序

首先设置 Tag 属性，目的是为了在编程时可以取得相关对象的句柄。此处对如下四个需要进行 Callback 编程的控制对象设置 Tag 属性，它们的 Tag 属性分别为：

- 左右两个接收数字输入的可编辑文本框（Edit）fahrenheittext、celsiustext；
- 接收鼠标操作的静态文本框 trigger；
- 指示温度的显示条的静态文本框 redline。

由于本例中事件较少，并且进行温度转换的处理方法上有相似之处，因此不再单独为每个事件编写独立的 Callback 程序，而是编写函数文件 temperature，通过调用函数 temperature 来对温度转换进行集中处理。在每个相关的控制对象中，在其 Callback 属性中写出相应的函数调用。具体如下：

- 左边的 Edit 对象 temperature fahrenheittext；
- 右边的 Edit 对象 temperature celsiustext；
- 接收鼠标操作的 text 对象 ButtonDownFcn 属性的 Callback 设为 temperature start。

以上操作如图 9-9 所示。

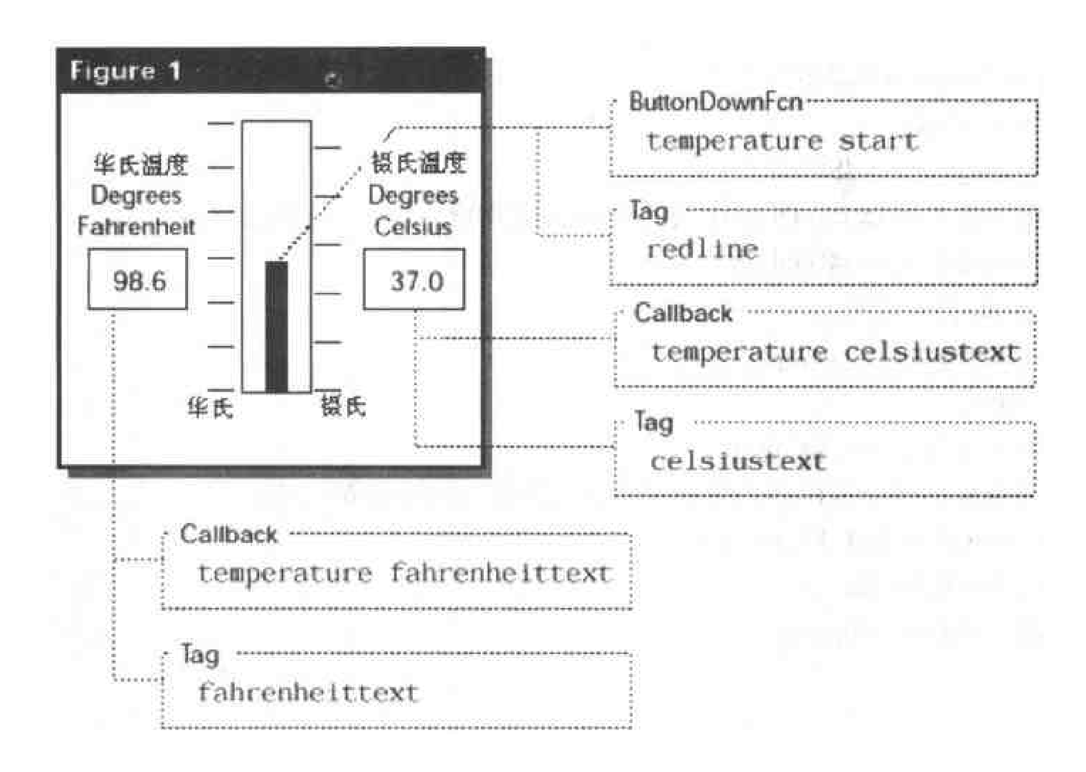


图 9-9　控制对象的属性设置

此图形用户界面的 temperature 函数子程序为：

```
function temperature(action)
%This is the callback function for the temperature conversion function
switch action
%如下的 start, move 和 stop 项用来激活红色的温度显示条
case 'start'
set(gcbf,'WindowButtonMotionFcn','temperature move')
set(gcbf,'WindowButtonUpFcn','temperature stop')
temperature move
case 'move'
currentPoint = get(gca,'CurrentPoint');
newY = currentPoint(3);
```

```
%限制温度的指示范围，此处限制华氏温度 fah 在-40 和 240 之间，并求算相应的摄氏温度 cels
fah = min(240,max(-40,newY));
cels = (fah-32)*(5/9);
%调用子程序 LocalSetDisplay 进行鼠标移动范围和可编辑文本框（Edit）的温度数值显示
LocalSetDisplay(fah,cels)
case 'stop'
set(gcbf,'WindowButtonMotionFcn','')
set(gcbf,'WindowButtonUpFcn','')
case 'fahrenheit'
fah = eval(get(gcbo,'String'));
%调用子程序 LocalSetDisplay 进行可编辑文本框（Edit）的温度数值显示
fah = min(240,max(-40,fah));
cels = (fah-32)*(5/9);
LocalSetDisplay(fah,cels)
case 'celsius'
cels = eval(get(gcbo,'String'));
%限制鼠标的移动范围和可编辑文本框（Edit）的温度数值显示
cels = min(120,max(-40,cels));
fah = cels*(9/5)+32;
LocalSetDisplay(fah,cels)
end
```

子程序 LocalSetDisplay(fah,cels)的内容

```
function LocalSetDisplay(fah,cels)
%设置文字显示的格式
pointerHandle = findobj(gcbf,'Tag','redline');
set(pointerHandle,'YData',[-40 fah])
fahHndl = findobj(gcbf,'Tag','fahrenheittext');
set(fahHndl,'String',sprintf('%3.1f',fah))
celsHndl = findobj(gcbf,'Tag','celsiustext');
set(celsHndl,'String',sprintf('%3.1f',cels))
```

9.2.3 麦克斯韦分布曲线

本节将利用气体分子运动的麦克斯韦速度分布律，求 27℃下氮分子运动的速度分布曲线，并求速度在 300m/s~500m/s 范围内分子所占的比例，讨论温度 **T** 和分子量 mu 对速度 v 分布曲线的影响。

1. 建立计算模型

首先应知道麦克斯韦分布律的公式为：

$$f = 4\pi(\frac{m}{2\pi kT})^{3/2} \cdot v^2 \cdot \exp(\frac{-mv^2}{2kT})$$

为研究单个参数的影响，本例将首先把麦克斯韦分布律编为一个函数子程序，以便重复调用，同时将常用的常数项也放在子程序中。本节将通过该例子说明如何用复杂的数学公式绘制曲线。

需要再次强调的是：子程序不得与主程序放在同一个 M 文件中，而只能将子程序单独做成 M 文件放在与主程序同一个工作路径中。

2. MATLAB 编程

根据建立的计算模型编写 M 文件 C11L5，内容如下：

● 首先建立用于主程序调用的计算麦克斯韦分布律的子程序 mxwl

```
%The subfunction mxwl of C11L5
%C11L5 利用麦克斯韦速度分布律求分子的速度分布曲线的子程序
function f=mxwl(T,mu,v)
%mu、v、T 分别是分子量、分子速度和气体的绝对温度
R=8.31;    %气体常数
k=1.381*10^(-23);       %波尔茨曼常数
NA=6.022*10^23;         %阿伏加德罗数
m=mu/NA                 %分子质量
f=4*pi*((m/2*pi*k*T)).^(3/2).*exp(-m*v.^2./....
    (2*k*T)).*v.*v;
```

● 然后编写主程序 C11L5 如下：

```
主程序
T=300;
mu=28e-3;               %给出 T 和 mu 的值
v=0:1500;               %调出自变量数组
y=mxwl(T,mu,v);         %调用子程序
plot(v,y)，hold on      %绘制分布曲线
 %给定速度范围，并画出速度分布律曲线所围的区域
v1=300:500;
y1=mxwl(T,mu,v1);
fill([v1,500,300],[y1,0,0],'r')
%调用函数 trapz( )求该范围的概率积分
trapz(y1)
hold on
%为了看出不同的 T 和 mu 对曲线形状的影响，再次给定 T 和 mu，在同一幅图中绘
制分布律曲线的图形
```

```
T=200;
mu=28e-3;
y=mxwl(T,mu,v);
plot(v,y)
T=300;
mu=2e-3;
y=mxwl(T,mu,v);
plot(v,y)
```

其运行结果如图 9-10 所示。

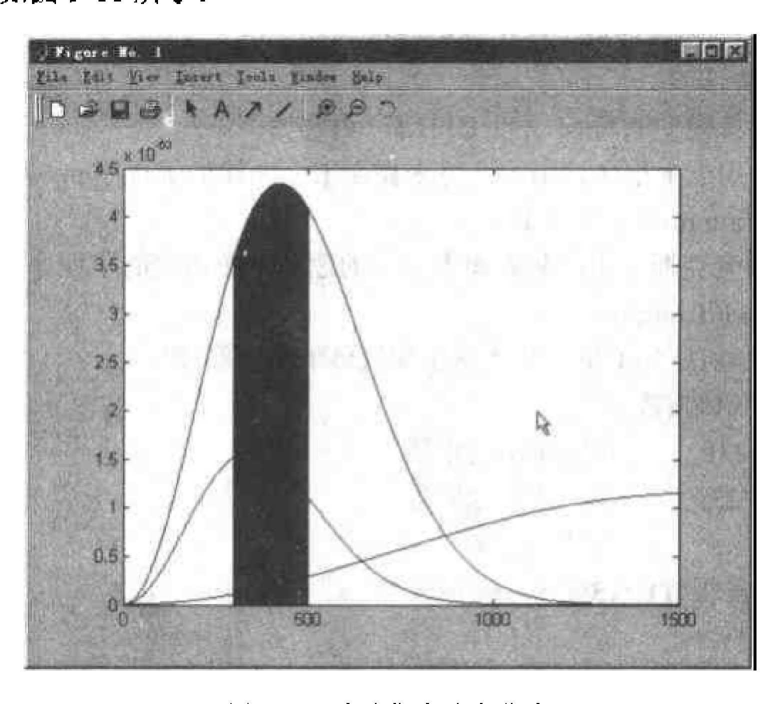


图 9-10　麦克斯韦分布曲线

9.3 信号系统

9.3.1 连续信号问题

在连续信号系统中，方波可以用相应频率的基波及其奇次谐波合成，这也是将它展开为正弦级数的出发点。本节将应用 MATLAB 来演示这一现象。

1. **建立计算模型**

一个以原点为奇对称中心的方波 $y(t)$可以用奇次正弦波的叠加来逼近，即

$$y(t)=\sin t+\frac{1}{3}\sin 3t+\frac{1}{5}\sin 5t+\cdots+\frac{1}{(2k-1)}\sin(2k-1)t+\cdots$$

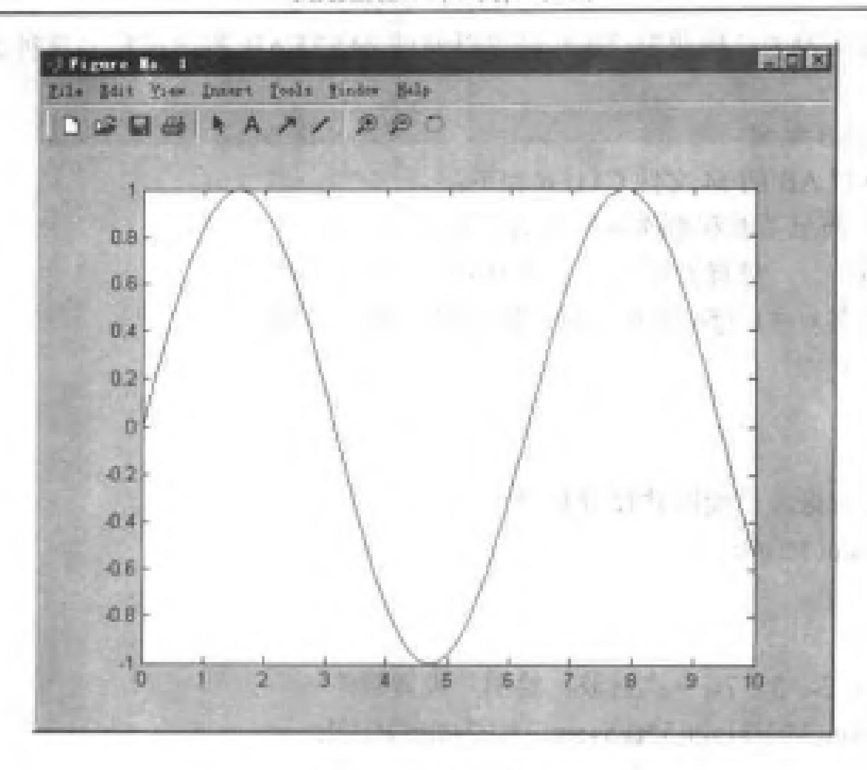
方波的宽度为π，周期为2π，则可以通过 MATLAB 程序来检验这种逼近的程度和特征。

2. MATLAB 编程

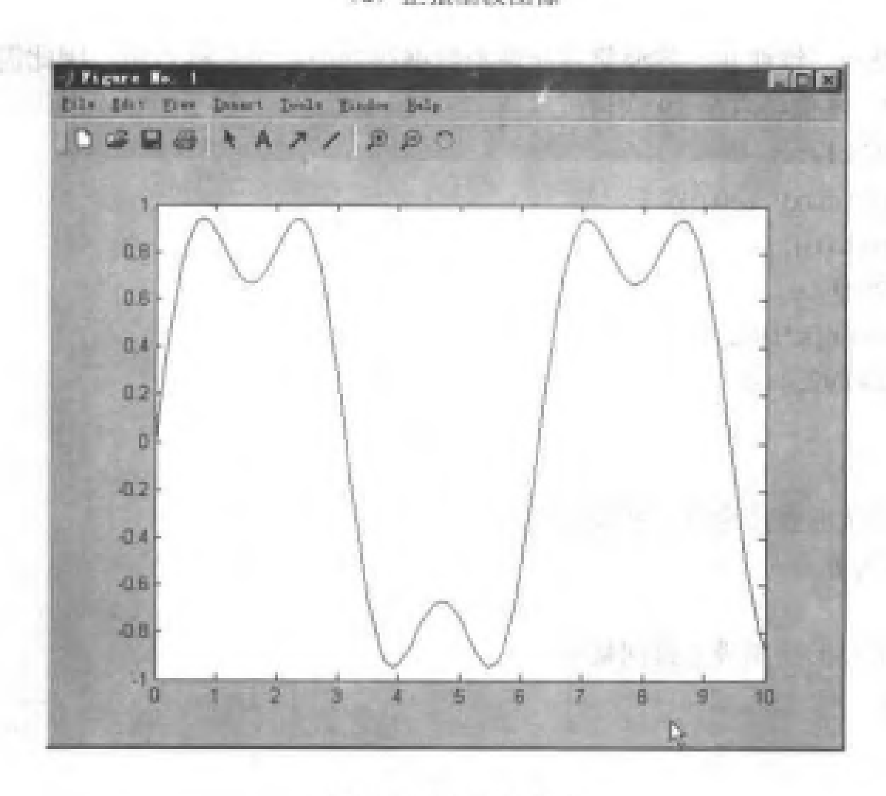
建立 MATLAB 的 M 文件 C11L6 如下：

```
%C11L6 演示基波和奇次谐波合成方波
t=0:0.1:10;        %首先设定一个有 101 个点的时间数组
%绘制频率 w=1（f=1/2π）的正弦基波，并设置暂停
y=sin(t);
plot(t,y)
pause
%叠加 3 次谐波，绘图并设置暂停
y=sin(t)+sin(3*t)/3;
plot(t,y)
pause
%叠加 1、3、5、7、9 次谐波，绘图并设置暂停
y=sin(t)+sin(3*t)/3+sin(5*t)/5+siu(7*t)/7+sin(9*t)/9;
plot(t,y)
pause
%为了绘制三维曲面，需要将各次波形数据存储为一个三维数组，因此需要重新定义
y，重新编程，本例将求至 19 次谐波
t=0:0.031:3.14;
y=zeros(10,max(size(t)));
x=zeros(size(t));
for k=1:2:19
    x=x+sin(k*t)/k;
    y((k+1)/2,:)=x;
end
pause
%将各个波形叠合绘出，并设置暂停
plot(y(1:9,:))
pause
%将各个波形绘制成三维网格图
mesh(y)
pause
clc
```

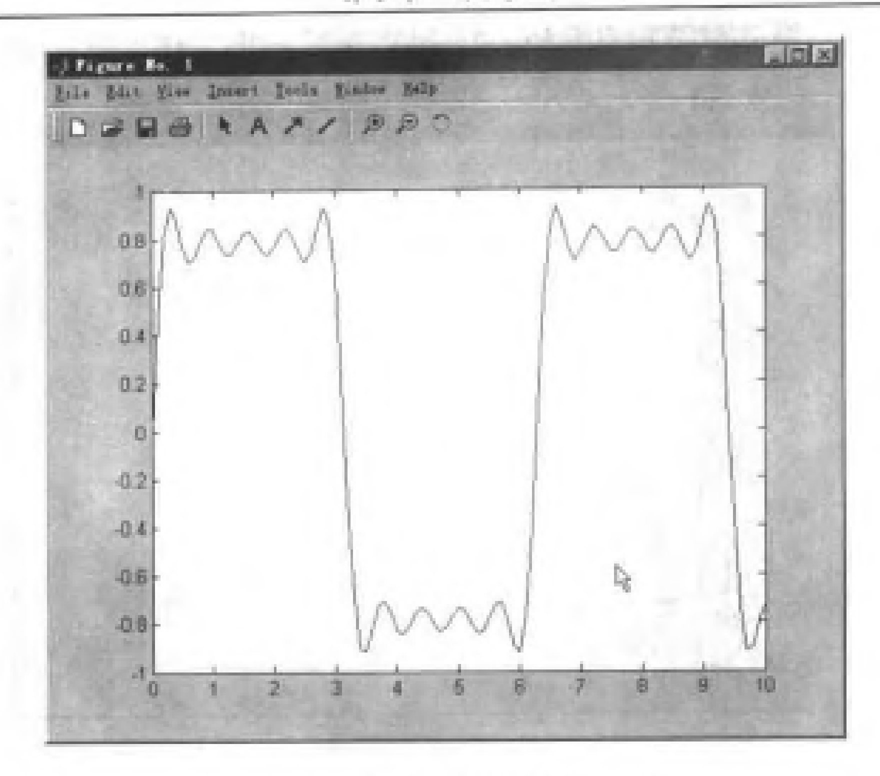
其执行结果如图 9-11 中（a）、（b）、（c）、（d）、（e）所示。



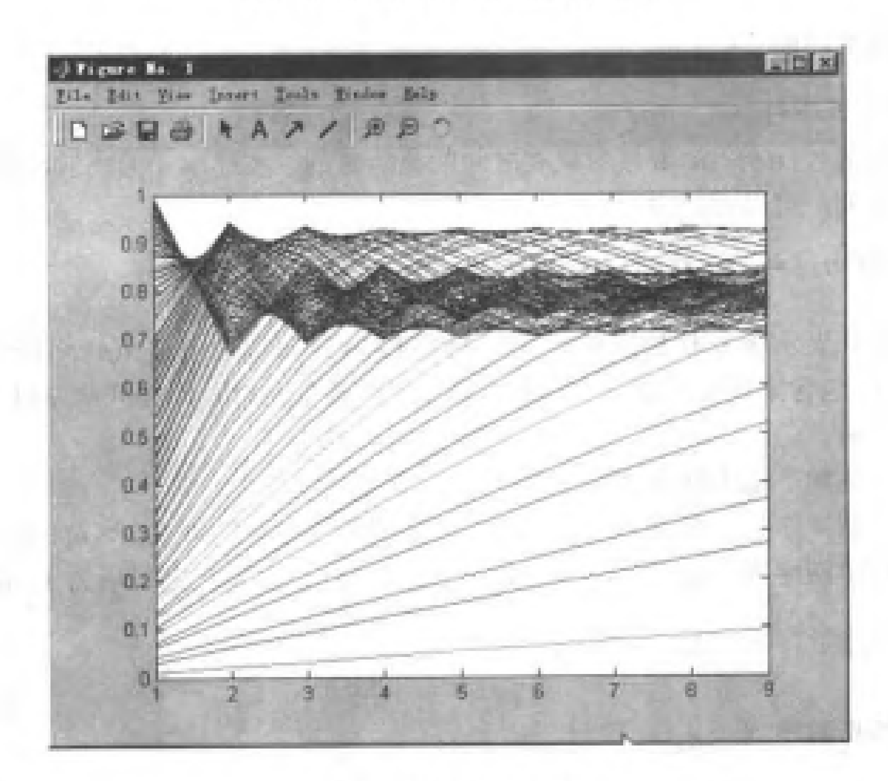
（a）正弦基波图像



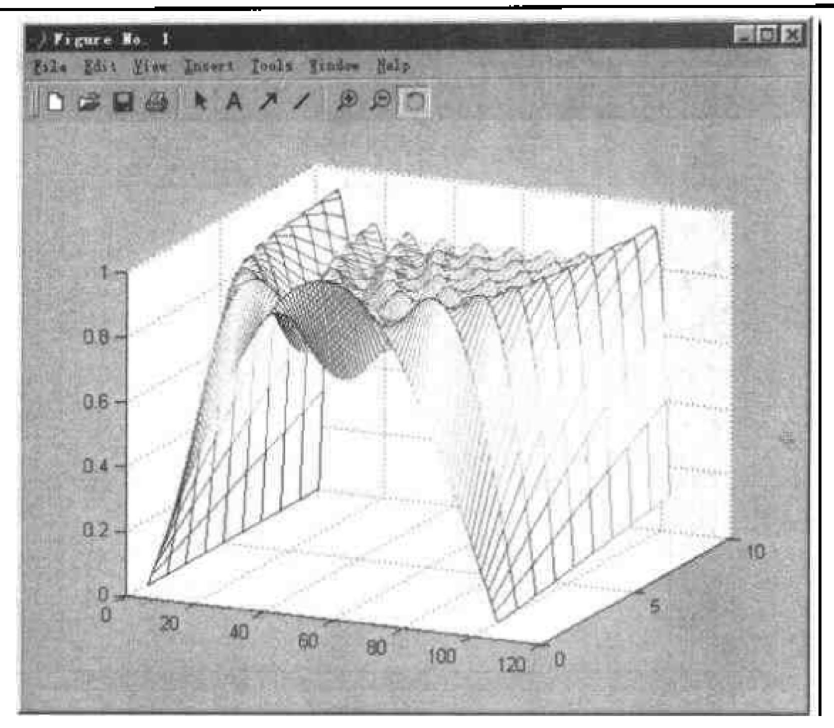
（b）3 次叠加谐波图像



（c）1、3、5、7、9 次叠加谐波图像



（d）19 次叠加谐波图像



（e）三维曲面网格图

图 9-11 三维网格图

9.3.2 离散信号问题

1. 绘制基本序列

对于离散信号 $x(n)$，n 取整数代表时间的离散时刻，实际 $x(n)$为离散时间信号，它是一个数字的序列，可以表述为：

$$x(n)=\{x(n)\}=\{\cdots,x(-1),\underset{\uparrow}{x(0)},x(1),\cdots\}$$

向上的箭头表示在 n=0 处的取样。在 MATLAB 中可以用向量 $x(n)$表示一个有限长度的序列，用 n 包含采样位置信息。当不需要采样位置信息时，即序列从 n=1 开始，可以只用 x 向量表示。

例如，产生如下基本脉冲序列：

- 单位脉冲序列　起点 n_0，终点 n_f，在 n_s 处有一单位脉冲（$n_0 \leqslant n_s \leqslant n_f$）；
- 单位阶跃序列　起点 n_0，终点 n_f，在 n_s 前为 0，在 n_s 处及以后为 1（$n_0 \leqslant n_s \leqslant n_f$）；
- 实数指数序列　$x_3=(0.9)^n$；
- 复数指数序列　$x_4=e^{(-0.2+0.3j)n}$。

根据上述基本序列表达式编写的 MATLAB 程序文件 C11L7.m 如下，注意，在产生单位阶跃序列时采用的逻辑关系方法，程序中调用函数 stem 来绘制离散序列。

```
clear
%C11L7 产生基本脉冲序列
```

```
%首先给定初值
no=0;
nf=10;
ns=3;
n1=no:nf;
%逻辑关系产生单位脉冲 x1 和阶跃序列 x2
x1=[zeros(1,ns-no),1,zeros(1,nf-ns)];
n2=no:nf;
x2=[zeros(1,ns-no),ones(1,nf-ns+1)];
%产生实数指数序列
n3=no:nf;
x3=(0.9).^n3;
%产生复数指数序列
n4=no:nf;
x4=exp((-0.2+0.3j)*n3);
subplot(2,2,1),stem(n1,x1);
xlabel('单位脉冲序列')
subplot(2,2,2),stem(n2,x2);
xlabel('单位阶跃序列')
subplot(2,2,3),stem(n3,x3);
xlabel('实指数序列')
%在图形窗口的第个四位置绘制两幅图，注意 subplot 的输入变元；同时调用 line 函数画横坐标
subplot(4,2,6),stem(n4,real(x4));
ylabel('实部')
subplot(4,2,8),stem(n4,imag(x4));
ylabel('虚部')
xlabel('复数序列')
line([0,10],[0,0])
```

程序的运行结果如图 9-12 所示。

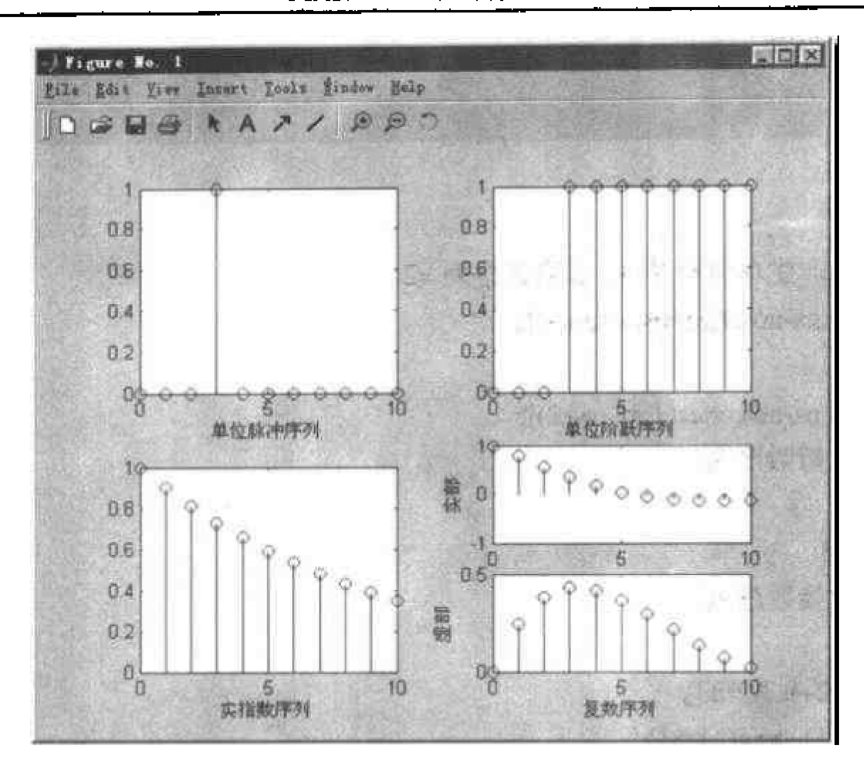


图 9-12 各种基本脉冲序列的波形图

2. 计算线性时不变系统差分方程

线性时不变系统的特性可以用差分方程来描述，如下所示：

$$a_1 y(n)+a_2 y(n-1)+\cdots+a_{n_a} y(n-n_a+1)=b_1 u(n)+b_2 u(n-1)+\cdots+b_{n_b} u(n-n_b+1)$$

因此，其左端和右端的系数向量可以充分地描述系统的特性。如果给定系统的输入，则系统的输出可以用 MATLAB 表示为：

$$a(1)\cdot y(n)=b(1)*u(n)+b(2)*u(n-1)+\cdots+b(nb)*u(n-nb+1)$$

$$-a(2)*y(n-1)-\cdots-a(na)*y(n-na+1)$$

令 $us=[u(n),\cdots,u(n-nb)]$； $ys=[y(n-1),\cdots,y(n-na+1)]$

则上式可以写成：

$$a(1)*y(n)=b*us'-a(2:na)*ys'$$

此方程可以采用递归的方法进行计算。

注意：由于 MATLAB 的变量下标只允许取正值，而公式需要指导 $n=1$ 之前的 y 和 u，程序中另设了两个变量 ym 和 um，相当于把 y 和 u 右移 $na-1$ 位，故 ym 和 um 的第 1 到 $na-1$ 位相当于 y 和 u 在起点之前的初值。在程序中随着计算点的右移，要随时生成相应于公式中的向量 us 和 ys。

根据以上说明编写 MATLAB 的 M 文件 C11L8 如下：

```
clear
a=input('差分方程左端系数向量 a=[a(1),a(2),…a(na)]=');
b=input('差分方程右端系数向量 b=[b(1),b(2),…b(na)]=');
u=input('输入信号序列 u=');
na=length(a);
nb=length(b);
nu=length(u);
s=['起算点前',int2str(na-1),'点 ym 的值=[m(1),…ym(na)]='];
%先预设 ym 序列的长度 na+nu，并预置 0，分配内存空间
ym=zeros(1,ua+nu-1);
%给 ym 序列的前 na 个点赋初值
ym(1:na-1)=input(s);
%建立 um 序列并赋初值
um=[zeros(1,na-1),u];
%以 ym 的起点为基准，生成 us 和 ys，调用差分方程递推求 ym
for n=na:na+nu-1
    ys=ym(n-1:-1:n-na+1);
    us=um(n-1:-1:n-nb);
    ym(n)=(b*us'-a(2:na)*ys')/a(1);
end
%将 ym 时间坐标左移 na-1 位，求 y
y=ym(na:na+nu-1);
stem(y)
grid on
%调用函数 line 画横坐标
line([0,60],[0,0])
```

在 MATLAB 的命令窗口中键入“C11L8”，运行程序，按照屏幕提示依次输入如下数值：

差分方程左端系数向量 a=[a(1),a(2),…a(na)]=[1,-1,0.9]

差分方程右端系数向量 b=[b(1),b(2),…b(na)]=[2,3]

输入信号序列 u=cos(0.5*[1:59])

起算点前两点 ym 的值=[m(1),…ym(na)]=[0,0]

程序的运行结果如图 9-13 所示。

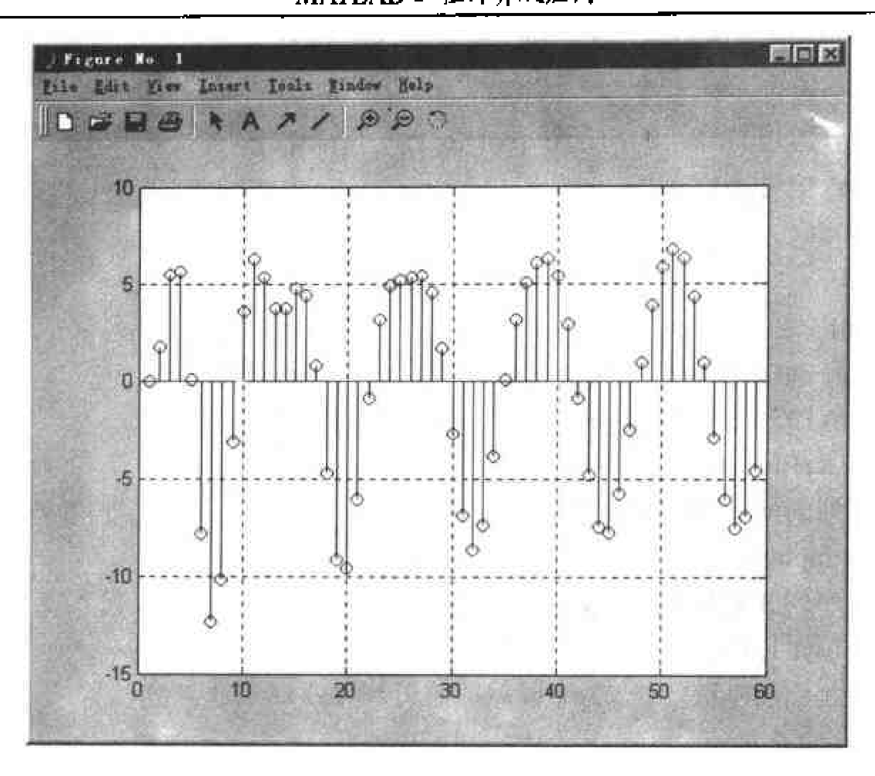


图 9-13 线性时不变系统差分方程的解

9.4 其他类问题

绘制给定函数的曲面图形。

给定函数 $\dfrac{x^2}{a^2}+\dfrac{y^2}{b^2}+\dfrac{z^2}{c^2}=d$，求绘函数表示的二次曲面，讨论参数 a、b、c 对形状的影响，分别画出图形。

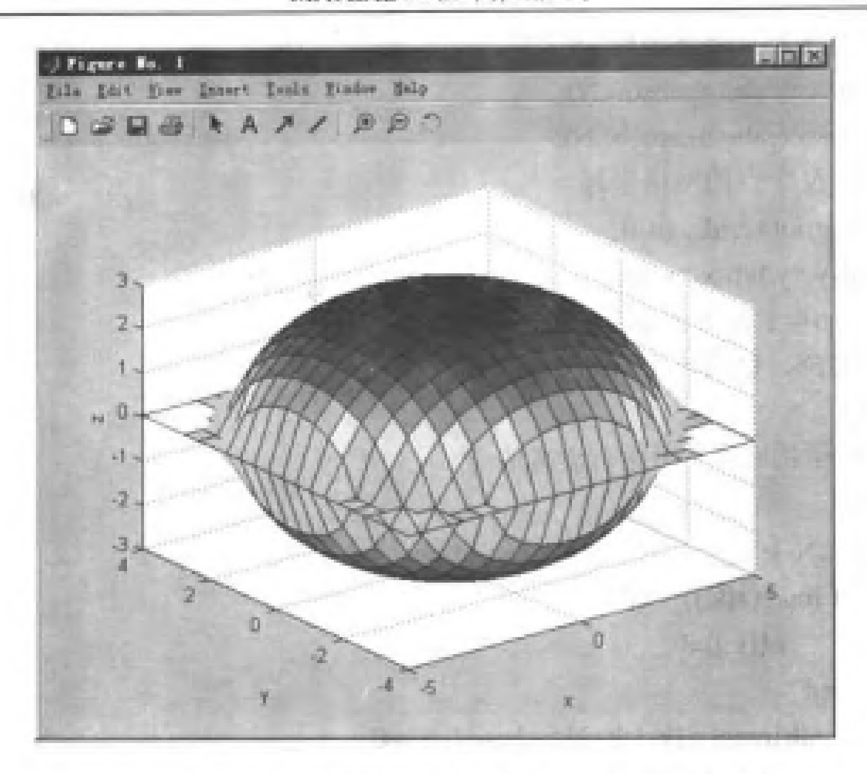
根据公式代表的数学模型绘制三维曲面图形，在编程时要注意在给定了 x、y 值求解 z 值时，如果有开方运算，将会出现虚数，如果是实数也有正负两个解。为了使虚数不出现在绘图中，本例将虚数全部转换为非数（NaN）。

建立 M 文件 C11L9 如下：

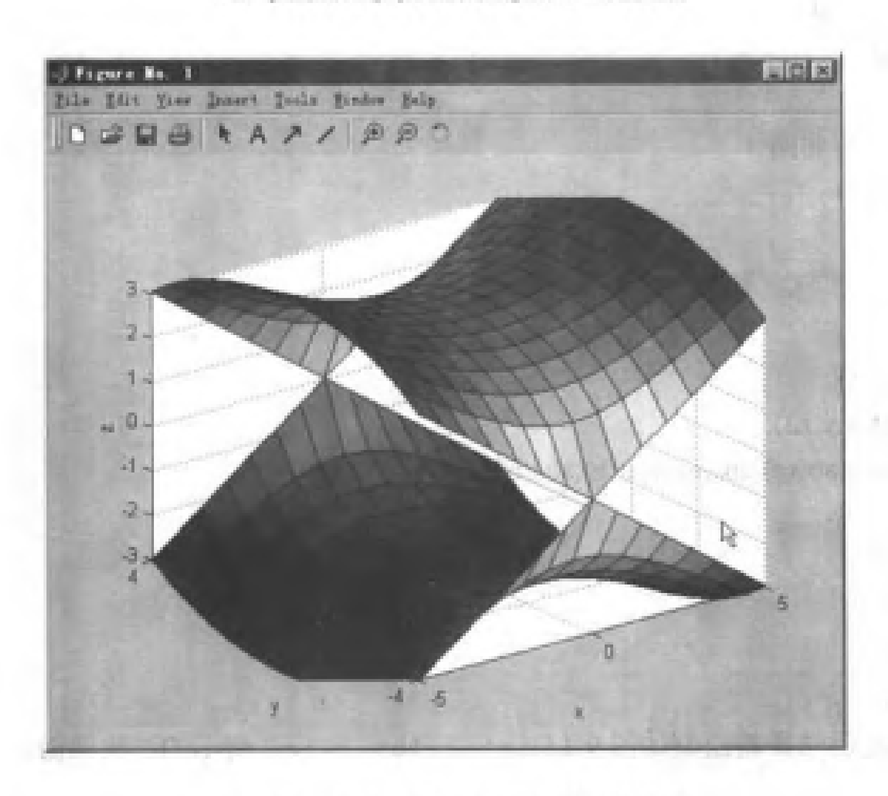
```
%C11L9 绘制二次曲面
%首先输入各种参数，并给出划分网格线的数目N
a=input('a=');
b=input('b=');
c=input('c=');
d=input('d=');
N=input('N=');
```

```
%分别建立 x 和 y 网格坐标
xgrid=linspace(-abs(a),abs(a),N);
ygrid=linspace(-abs(b),abs(b),N);
%确定 N×N 个点的网格坐标
[x,y]=meshgrid(xgrid,ygrid);
z=c*sqrt(d-y.*y/b/b-x.*x/a/a);
u=1;     %u=1 表示 z 要取正负值
%取 z 的实部，赋值给 z1
z1=real(z);
%采用 for 结构取消 z 中含有虚数的点
for k=2:N-1
    for j=2:N-1
        if imag(z(k,j))~=0
            z1(k,j)=0;
        end
        if all(imag(z([k-1:k+1],[j-1:j+1])))~=0
            z1(k,j)=NaN;
        end
    end
end
%绘制三维曲面
surf(x,y,z1)
hold on
%u=1 时加画负半面，并加负向坐标轴
if u= =1
    z2=-z1;
    surf(x,y,z2);
    axis([-abs(a),abs(a),-abs(b),abs(b),-abs(c),abs(c)]);
end
xlabel('x')
ylabel('y')
zlabel('z')
hold off
```

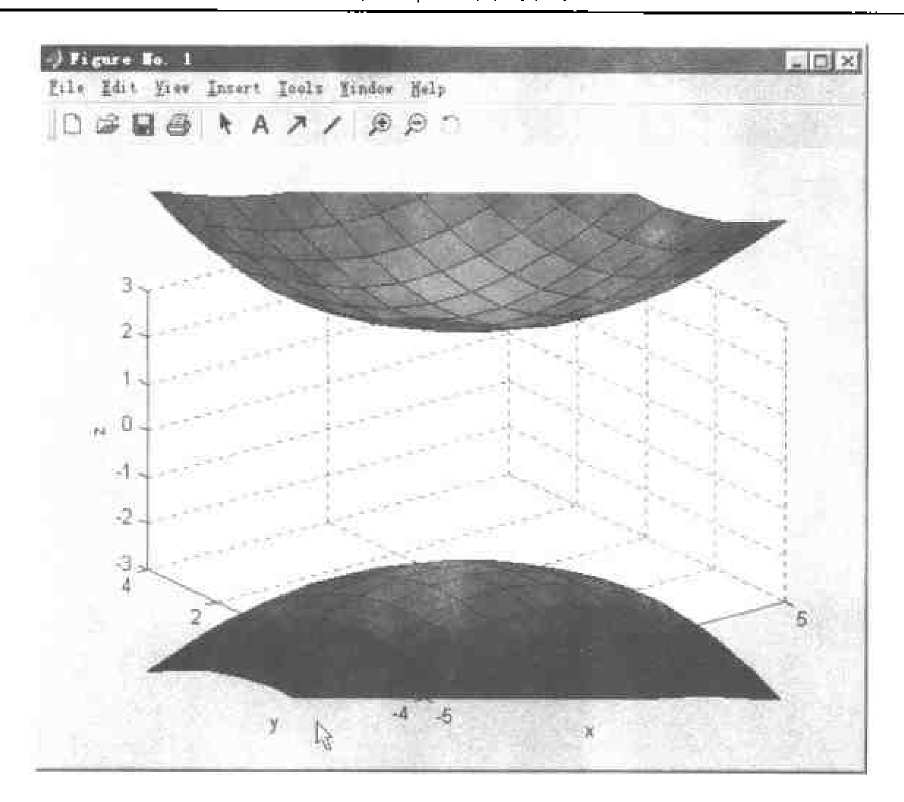
程序的运行结果将绘制如图 9-14（a）、（b）、（c）所示的二次曲面。



（a）[a,b,c,d,N]=[5,4,3,1,20]时的二次曲面



（b）[a,b,c,d,N]=[5i,4,3,1,15]时的二次曲面



（c）$[a,b,c,d,N]=[5i,4i,3,1,10]$时的二次曲面

图9-14　多种不同的二次曲面图

9.5 小　结

作为全书的结束，本章给出了MATLAB在一些工科课程及教学中的实际应用，便于读者能够对MATLAB有更好的理解和掌握。主要包括如下几部分的内容：

（1）单自由度体系有阻尼振动问题。

（2）双自由度可解耦系统的振动问题。

（3）热力学和分子分布问题实例。

（4）信号系统的连续和离散问题实例。

（5）绘制函数的图像问题实例。